INTELLIGENCE ARTIFICIELLE SUR SINCLAIR QL

Keith & Steven Brain

Edimicro

INTELLIGENCE ARTIFICIELLE SUR SINCLAIR QL

Faites réfléchir votre micro

INTELLIGENCE ARTIFICIELLE SUR SINCLAIR QL

Faites réfléchir votre micro

par

Keith et Steven BRAIN

Edimicro

Traduction autorisée de l'ouvrage publié en langue anglaise.
Original title : « Artificial Intelligence on the Sinclair QL ».
Copyright © 1984 Keith and Steven Brain.

First published in English 1985 by : Sunshine Books.
12-13, Little Newport Street, London WC2R 3LD.

Traduction et adaptation française : Hervé Riondel, Eric Tenin.

Couverture : Thierry Duval.

Imprimé en France. Droits mondiaux réservés.

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective", et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite" (alinéa 1er de l'article 40). » « Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. »

© 1985 F.D.S./Edimicro pour la traduction française
© 1985 F.D.S./Edimicro pour l'adaptation à la version française du QL
ISBN 2-904457-36-4

EDIMICRO
DÉPARTEMENT ÉDITIONS DE F.D.S. SARL
121/127, avenue d'Italie, 75013 Paris

Table des matières

INTRODUCTION

CHAPITRE 1 : **L'intelligence artificielle** 1

L'imaginaire 1
La réalité 3

CHAPITRE 2 : **L'exécution des ordres** 7

Un petit problème 8
Ordres invalides 12
Un peu d'action 14
Utilisation des procédures de direction 16
Une méthode plus souple 17
Extension du vocabulaire 20
Suppression de la redondance 21
Utilisation de commandes abrégées 24
Egalité sur la première lettre 24
Des commandes enchaînées 25

CHAPITRE 3 : **La compréhension du langage naturel** 31

Traiter les phrases 32
Décomposer la phrase 32
La ponctuation 41
Une méthode de recherche itérative 44
Correspondance partielle 46
Mettre les choses en ordre 48

CHAPITRE 4 : **La construction des réponses** 53

Des réponses sensées 53
Une autre dimension 55
Des réponses plus élaborées 58
Correspondance 61
Réponses 62
D'autres sujets 63

Choisir un sujet selon le contexte 64
Insérer le sujet dans la phrase 66
Changement de temps 68

CHAPITRE 5 : **Les systèmes experts** 69

Des branchements pour sortir 72
Vers une solution 75
L'approche parallèle 77
De mieux en mieux 81
Encore mieux avec des chaînes de bits 82

CHAPITRE 6 : **Un système expert qui enrichit ses connaissances** 87

Duquel s'agit-il ? 89
Généralisation 94
Digestion automatique des données 106
Ça tourne 108
La conservation de notre expert 109

CHAPITRE 7 : **Des comparaisons approchées** 111

Routine de codage 112
La mise au point des détails 118
Utilisation 120
Egalité partielle 122

CHAPITRE 8 : **La reconnaissance des formes** 125

Elimination d'une branche 127
Un élagage 127

CHAPITRE 9 : **Un professeur intelligent** 139

Questions et réponses 139
Si ce n'est pas un nombre ? 140
D'autres opérations 141
Division par zéro 142
Elimination des décimales 142
Mémorisation du résultat 142
Combien de questions ? 143
Suivi de la progression 144
Degré de difficulté 146
D'autres types de questions 148

CHAPITRE 10 : **Des souris et des hommes** 149

Plaçons le décor 149
Construction du labyrinthe 150
Trouver le chemin 152
Est-ce gagné ? 154

Quel chemin ? 155
Les coins 155
Moins de tests 157
Les bifurcations 158
Les retours en arrière 161
Où ne vont pas les souris 162
La vitesse de croisière 164

CHAPITRE 11 : **Utilisation intelligente du programme Archive** 167

Recherche d'une correspondance 169
Recherches spécifiques 170
Isolement de certains enregistrements 172
Mettre les choses en ordre 173
Correspondances partielles 174
Utilisation de PROCédures 174
Une (inter) face plus plaisante 177

CHAPITRE 12 : **Un vendeur expert** 185

Conversation 188
Décisions 191

Introduction

L'intelligence artificielle est certainement une étape capitale dans le développement des ordinateurs, qui aura des conséquences importantes sur notre vie dans les prochaines décades. L'objectif principal de ce livre est de présenter à un lecteur peu averti quelques concepts utilisés en intelligence artificielle, et de lui montrer comment développer de petites routines sur le QL, qu'il pourra incorporer dans ses propres programmes. Cet ouvrage ne suppose qu'une connaissance superficielle de la programmation, et part des principes de base. Nous croyons en effet que c'est essentiel pour comprendre les problèmes d'intelligence artificielle, et les résoudre.

L'idée de base de ce livre est de développer des idées et des routines appropriées, pas à pas, en analysant et en comparant les différentes possibilités. Plutôt que de vous donner une série de programmes complets, nous vous encourageons à essayer les différentes approches pour vous faire voir les résultats des différentes techniques. Des organigrammes détaillés sont donnés, pour la plupart des programmes.

Bien que l'accent principal porte sur l'aspect « intelligence artificielle », nous en avons profité pour exploiter les points forts du superBASIC, tels que PROCédures, FONction, Fenêtre.

Vous pourrez remarquer que par endroit certaines lignes sont redondantes. Elles ont été laissées volontairement, dans un but de clarté. Autant que possible, on a évité d'avoir à retaper des lignes, en portant plutôt des modifications de routines. Certaines PROCédures sont communes à plusieurs chapitres, et nous vous rappelons qu'elles peuvent être sauvegardées séparément sur microdrive, puis transférées dans les différents programmes avec la commande MERGE.

Tous les listings du livre sont présentés de façon à montrer clairement le déroulement du programme, aussi ils n'apparaîtront pas de la même façon à l'écran. Dans beaucoup de cas des espaces et des parenthèses ont été utilisés pour rendre les listings plus lisibles ; mais, attention : certains sont indispensables, ne soyez pas tenté de les faire tous disparaître.

Tous les programmes ont été rigoureusement testés et les listings vérifiés avec beaucoup de soins, aussi nous espérons que vous n'y trouverez pas d'erreurs. Il est triste de constater que les erreurs sont le plus souvent le résultat de « fote de frappe » de l'utilisateur. Points-virgules et virgules peuvent avoir l'air d'être sans signification, mais leur absence a des conséquences importantes.

L'intelligence artificielle devient chaque jour plus importante, et nous espérons que ce livre vous donnera un aperçu utile dans ce domaine. Qui sait : si vous approfondissiez le sujet, vous pourriez persuader votre machine de lire elle-même notre prochain livre !

Une fois encore, merci à Liz d'avoir résisté à la tentation de jeter toutes ces feuilles par la fenêtre ou d'en retapisser la pièce.

Keith et Steven Brain
Groeswen, juillet 1984.

1

L'intelligence artificielle

L'IMAGINAIRE

Pendant des générations, des auteurs de science-fiction ont envisagé le développement de machines intelligentes qui pourraient réaliser beaucoup de fonctions propres à l'homme, et même le dépasser dans certains domaines, et pour le grand public l'intelligence artificielle est associée à ces images.

L'aspect usuel d'un robot est celui d'une machine intelligente de forme humaine, qui est capable d'exécuter des instructions données d'une manière très générale. La plupart des gens ont des notions vagues dans les domaines techniques, et dans les histoires anciennes ces robots avaient très mauvaise presse, étant cantonnés dans les rôles de mauvais sujet, totalement inconscient et invulnérable.

Isaac Asimov, qui voyait loin, tissa une interminable série d'histoires autour de son concept de robot "positronique", et fut probablement le premier auteur à essayer de coller à la réalité. Il écrivit son fameux "Trois lois de Robotique", qui fixent les règles fondamentales à partir desquelles doivent être construites les machines capables d'autonomie d'action. Mais il est intéressant de remarquer qu'il ne pouvait imaginer quand la race humaine accepterait la présence de tels robots sur terre. "La guerre des étoiles" a amené les robots spécialisés R2D2 et C3PO, mais beaucoup

de leurs caractéristiques nous semblent étranges: il y a peut-être une "Union inter-planétaire des robots" et une frontière qui empêche les communications directes entre les hommes et R2D2.

Des ordinateurs évidemment intelligents apparaîssent sous la forme de boîtes sans bras ni jambe, lançant obligatoirement des flashs lumineux. Les échanges se font évidemment par la parole, mais la voix métallique d'autrefois n'est plus à la mode, elle a pris une certaine personnalité. Que toutes les boîtes se ressemblent c'est peut-être une bonne idée, mais que toutes les voix ne soient pas celle de Zorro dans "Terrahawks"!

Le KITT de Michael Knight se présente comme une machine dotée de raison avec laquelle on peut dialoguer, on la préfère au SLAVE huileux ou à l'odieux ORAC des "Sept de Blake". ORAC semble avoir accumulé une montagne de mépris dans sa petite boîte. D'autres écrivains ont évalué les difficultés qui peuvent se présenter lorsque la personnalité d'une machine est trop proche de celle de l'homme. Dans "2001: Odyssée de l'espace" l'ordinateur surdoué HAL a une dépression nerveuse lorsqu'il se trouve devant trop de responsabilités. Dans "Le restaurant du bout du monde", la valeur du robot transporteur SIRIUS s'effondre car, prévoyant l'avenir, il refuse d'avancer lorsqu'il se rend compte qu'il risque de se faire démolir; quant au "Synthétiseur de boissons nutritives" il a manifestement été conçu par le service des approvisionnements des chemins de fer britanniques, car pratiquement dans tous les cas il donne une sorte de thé. Plus récemment la silhouette holographique de "Automan" a montré des capacités amusantes dans son combat contre le crime, bien que sa programmation en ce qui concerne les sentiments féminins semble présenter des lacunes.

D'autres thèmes plus ennuyeux sont apparus récemment. La caractéristique la plus significative de "Wargames" n'est pas que quelqu'un réus-

sisse à percer JOSHUA (l'ordinateur de la Défense des USA), mais le fait que lorsqu'une guerre thermo-nucléaire est déclenchée, la machine ne peut s'arrêter avant qu'il y ait un gagnant. Dans le "Projet Forbin", les ordinateurs américains et russes s'entendent pour dire que les hommes manquent de sagesse.

LA REALITE

La notion d'intelligence pour une machine suscite une violente querelle d'experts. La définition la plus généralement acceptée est celle qui a été proposée par Alan Turing à la fin des années quarante, à l'époque où les ordinateurs avaient la taille d'une maison. Plutôt que de poser une série de critères à remplir, il a pris du recul vis-à-vis du problème. Il dit que la plupart des gens acceptent que les hommes sont intelligents et que, si quelqu'un ne peut pas reconnaître s'il a affaire à un homme ou à un ordinateur, il doit admettre que celui ci est aussi doté d'intelligence. Ceci est le fondement du fameux test de Turing, qui consiste à mettre un opérateur en relation avec une autre entité par l'intermédiaire d'un clavier, et à essayer de faire dire à celui-ci si c'est une machine ou un homme - ce qui n'est pas commode!

On a raconté beaucoup d'histoires sur ce test, la meilleure est celle d'un demandeur d'emploi qui est placé devant un clavier et livré à lui-même. Il se rend compte de l'importance de ce test pour la suite de sa carrière, et se bat vaillamment pour découvrir la vérité, mais sans succès. Au bout d'un moment, la personne qui le reçoit, revient et lui dit : "Bien joué, la machine n'a pas pu déterminer si vous êtes un homme, vous êtes la personne qu'il nous faut comme inspecteur des impôts!

Tout le monde a entendu à la télévision que l'usage des techniques d'aide à la conception par ordinateur étaient très courantes, et que

les robots industriels étaient presque seuls devant les chaînes de montage des voitures (jusqu'aux autocollants sur les vitres qui indiquent "Dessinée par un ordinateur, construite par un robot, et conduite par un idiot"). En fait ces robots industriels ont fort peu d'intelligence, ils ne font que suivre un programme, et ont fort peu de décisions à prendre. Le robot qui fait la peinture a appris les gestes d'un opérateur et il est incapable de peindre un nouvel objet sans une autre intervention de celui-ci.

Par ailleurs la nouvelle génération de robots a des senseurs et un logiciel beaucoup plus sophistiqués qui lui permettent de déterminer la forme, la couleur et la texture des objets, et de prendre des décisions beaucoup plus rationnelles. Tous ceux qui ont vu la légendaire "micromouse" courir tout droit jusqu'au centre du labyrinthe ne s'étonneront pas de notre foi dans le robot intelligent du futur, même si celui-ci n'a ni bras ni jambes, ce qui est sans importance.

Un autre domaine important où l'intelligence artificielle est d'usage courant est celui des systèmes experts; beaucoup d'entre eux sont capables de faire aussi bien sinon mieux que des hommes, dans le domaine des prévisions météo par exemple. On rencontre ces systèmes experts dans beaucoup de domaines et en particulier dans celui du diagnostic et du traitement médical. Les médecins n'ont pas à s'inquiéter, car les ordinateurs ne savent pas ausculter.

Une des principaux obstacles à la diffusion des ordinateurs est l'ignorance et l'entêtement des utilisateurs qui, à la rigueur liront les instructions, mais attendront de la machine qu'elle prenne en compte les spécificités de leur problème. L'usage du langage naturel se développe rapidement, et les ordinateurs de la "cinquième génération" seront beaucoup plus conviviaux.

La plupart des travaux sérieux en

intelligence artificielle sont développés dans des langages plus appropriés, comme LISP ou PROLOG, mais malheureusement ceux-ci sont inintelligibles pour l'usager moyen! Les routines du SuperBASIC qui suivent ne vous donneront pas la clé de la domination du monde, mais elles vous donneront une bonne appréciation des possibilités et des problèmes posés par l'intelligence artificielle. Comme tous les spécialistes, les experts dans ce domaine ont leur propre jargon pour impressionner les novices. Cependant, comme ce livre s'adresse à Monsieur Tout-le-Monde, nous avons délibérement évité l'emploi de ce jargon chaque fois que cela était possible, car nous pensons qu'il jette la confusion dans les esprits.

2

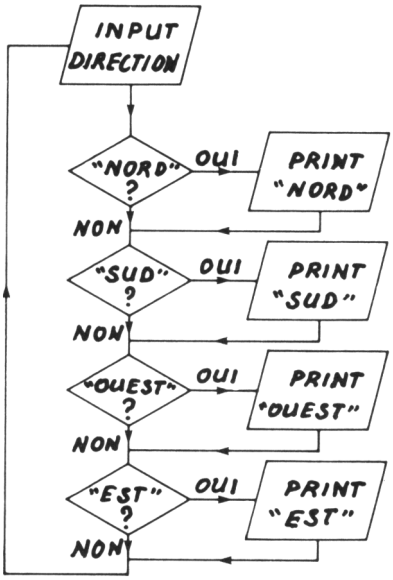
L'exécution des ordres

Comme votre ordinateur n'a aucune intelligence, vous ne pouvez dialoguer avec lui qu'à un niveau élémentaire de façon très formelle. Nous vous montrerons plus loin comment surmonter cette barrière du langage, mais commençons par apprendre à marcher avant de courir. Les premiers pas consistent à fixer une série d'ordres auxquels correspondent des réponses pré-établies.

Nous allons commencer par examiner le problème de faire comprendre à votre ordinateur les directions d'une boussole. Au premier abord, la façon la plus simple de le programmer consiste à utiliser une boucle REPeat qui demande une entrée (INPUT) à l'utilisateur, et contient une ligne IF-THEN pour chacune des possibilités. (Organigramme 2.1.)

Nous allons utiliser deux PROCedures, qui seront peu à peu affinées tout au long du chapitre. Pour le moment START efface simplement l'écran, tandis que VERS compare l'entrée avec les quatre mots-clés.

```
   10 START
   20 VERS

10000 DEFine PROCedure VERS
10010   CLS
10990 END DEFine START
```




```
11000 DEFine PROCedure VERS
11010   REPeat DIRECTION
11020      UNDER 0:PRINT \"DIRECTION ?"
11030      INPUT CHAINE$ : UNDER 1
11100      IF CHAINE$="NORD" THEN PRINT "NORD"
11110      IF CHAINE$="SUD" THEN PRINT "SUD"
11120      IF CHAINE$="OUEST" THEN PRINT "OUEST"
11130      IF CHAINE$="EST" THEN PRINT "EST"
11980   END REPeat DIRECTION
11990 END DEFine VERS
```

Organigramme 2.1: Les directions de la boussole.

Pour distinguer clairement l'INPUT de la réponse du QL, nous avons souligné cette réponse, et le slash inverse force le saut à la ligne.

UN PETIT PROBLEME

En testant cette routine vous allez tomber sur un petit problème courant: l'ordinateur ne trouve l'égalité que pour les lettres majuscules.

Ainsi "NORD" est égal à "NORD", "nord" à "nord", mais pas à "NORD".

La façon la plus simple est de tester si le premier caractère de CHAINE$ est une majuscule. Toutes les majuscules ont un code inférieur à 91, on peut donc envoyer un message d'avertissement s'il est nécessaire d'appuyer sur ALPHA.

```
11040     IF CODE(CHAINE$)>90 THEN
11050       CSIZE 2,1 :PRINT "appuyez sur ALP
HA!": CSIZE 1,0
11060       NEXT DIRECTION
11070     END IF
```

(En SuperBASIC il n'est pas nécessaire de spécifier la première lettre, CHAINE$(1), car CODE sauf indication contraire retourne la valeur du premier caractère d'une chaîne.)

Une approche plus élaborée consiste à demander à l'ordinateur de transformer automatiquement tous les caractères entrés avec une casse particulière. (Majuscule égale casse haute). Pour comprendre le fonctionnement, il faut regarder le tableau de représentation binaire des caractères. (Tableau 2.1.) Vous remarquerez que la seule différence entre une lettre du haut et la lettre correspondante du bas porte sur le bit 6, qui vaut 0 pour les majuscules, et 1 pour les minuscules.

Pour forcer à la fois majuscules et minuscules en minuscules, il faut positionner le bit 6 à 1. Pour l'inverse il faut positionner le bit 32 à 0.

Pour positionner le bit 6 il suffit de faire l'opération OU (OR) sur le code du caractère.

```
'A'=65                    01000001
OR 32                     00100000
                          --------
                          01100001 = 97 ('a')

'a'=97                    01100001
OR 32                     00100000
                          --------
                          01000001 = 97 ('a')
```

Pour mettre le bit 6 à 0, il suffit de faire l'opération NON (NOT) sur le code du caractère. Ceci le met à 0 quelle que soit sa valeur initiale.

```
'A'=65                          01000001
NOT 32                          00100000
                                --------
                                01000001 = 65 ('A')

'a'=97                          01000001
NOT 32                          00100000
                                --------
                                01000001 = 65 ('A')
```

Bit	8	7	6	5	4	3	2	1
Valeur	128	64	32	16	8	4	2	1
A	0	1	0	0	0	0	0	1
B	0	1	0	0	0	0	1	0
Y	0	1	0	1	1	0	0	1
Z	0	1	0	1	1	0	1	0
a	0	1	1	0	0	0	0	1
b	0	1	1	0	0	0	1	0
y	0	1	1	1	1	0	0	1
z	0	1	1	1	1	0	1	0

Tableau 2.1: Représentation binaire des caractères.

Nous allons définir une FONction (SAISI$) qui effectuera l'une ou l'autre de ces conversions. Elle n'utilise que l'opération OU du QL étant donné qu'une seule FONction suffit pour la conversion dans les deux sens (en lui passant un paramètre).

La FONction SAISI$ est appelée à la ligne 11030 qui remplace la précédente.

Le paramètre passé (cs) est soit 0, pour la conversion en minuscules, soit 1, pour la conversion en majuscules.

```
11030 CHAINE$=SAISIS$(1):UNDER 1
11040 REMark ANNULEE
11050 REMark ANNULEE
11060 REMark ANNULEE
11070 REMark ANNULEE
```

L'entrée (INPUT i$) s'effectue comme auparavant,mais à l'intérieur de la boucle SAISIS-CAR, chaque caractère est passé en revue et comparé bit à bit par l'intermédiaire d'une opération ou (||) avec 32. Cela force le bit 6 à 1. On notera cependant que la valeur de l'opération 32* cs (le paramètre passé) est tout de même soustrait du résultat. Si cs vaut zéro, cette opération sera sans effet et il en résulte des minuscules.En revanche si cs vaut 1, 32 sera soustrait, et cela agira alors sur le bit 6 et il en résultera de ce fait des majuscules. Lorsque la fin de la chaîne est atteinte (n=LEN (i$)), on sort alors de la boucle SAISIS-CAR et le résultat est RETourné à la ligne d'appel. (Organigramme 2.2.)

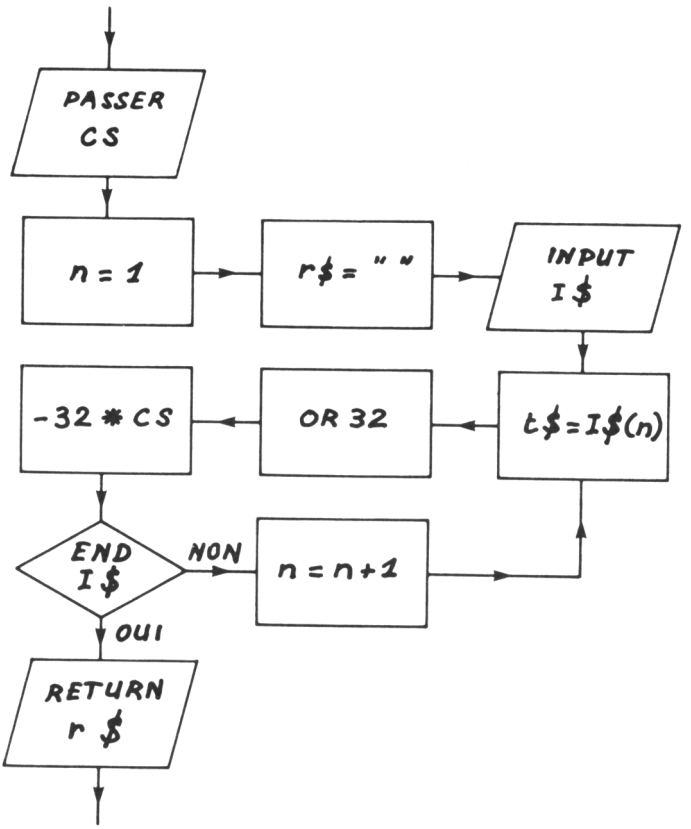
```
20000 DEFine FuNction SAISIS$(cs)
20010   LOCal i$,n,t$,r$
20020      n=1
20030      r$=""
20040      INPUT i$
20050        REPeat SAISIS_CAR
20060          t$=i$(n)
20070          r$=r$ & CHR$((CODE(t$) || 32)-
(32*cs))
20080            IF n=LEN(i$) THEN EXIT SAISI
S_CAR
20090          n=n+1
20100        END REPeat SAISIS_CAR
20110      RETurn r$
20120 END DEFine SAISIS$
```

Notez que toutes les variables utilisées sont LOCales et propres à la fonction, et sont en minuscules pour les distinguer des variables globales.

La routine modifiera tous les caractères, ce qui peut poser des problèmes, par exemple un espace (CHR$(32)) sera changé en CHR$(0) qui n'est pas visualisable! On peut éviter la conversion des espaces et des numériques en limitant la conversion à certains codes. Une façon simple de faire cette restriction à des

caractères de code supérieur à 32 est de multiplier (32*cs) par NOT CODE(t$) 65. Si le code de t$ est inférieur à 65, (32*cs) ne sera pas soustrait, et le caractère sera inchangé.

```
20070      r$=r$ & CHR$((CODE(t$) || 32)-(32*
cs) * NOT CODE(t$)<65)
```



Organigramme 2.2: PROCedure SAISIS/$.

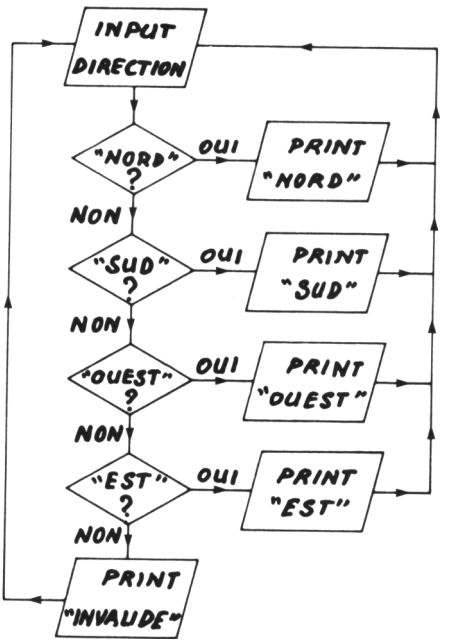
ORDRES INVALIDES

Si vous tapez autre chose que les quatre mots-clés, rien n'apparaîtra à l'écran, sinon l'attente d'une nouvelle entrée. Il serait préférable que votre ordinateur indique clairement que la commande est invalide. Vous pourriez ajouter un test indiquant qu'aucune commande valide n'a été trouvée, mais cela serait interminable, voire même impossible si la liste des mots-clés était plus longue.

```
11140      IF CHAINE$<>"NORD" AND CHAINE$<>"S
UD" AND CHAINE$<>"EST" AND CHAINE$<>"OUEST" T
HEN
11150       CSIZE 1,1:PRINT "ORDRE INVALIDE":
 CSIZE 1,0
11160      END IF
```

En revanche, lorsqu'un mot-clé a été reconnu, il suffit d'ajouter une instruction NEXT à la fin de chaque ligne de test (IF THEN) pour revenir directement à la ligne où l'entrée est demandée. Si aucun des tests n'est vérifié, le programme arrive directement à la ligne 11150 et le message d'avertissement vient s'afficher. De toute façon, il est bien plus logique d'ajouter cette instruction NEXT et de sauter ainsi directement à la ligne de l'entrée, car cela évite d'effectuer une longue suite de tests inutiles étant donné qu'un mot-clé a nécessairement été trouvé.

```
11100      IF CHAINE$="NORD" THEN PRINT "NORD"
:NEXT DIRECTION
11110      IF CHAINE$="SUD" THEN PRINT "SUD":N
EXT DIRECTION
11120      IF CHAINE$="OUEST" THEN PRINT "OUES
T": NEXT DIRECTION
11130      IF CHAINE$="EST" THEN PRINT "EST":N
EXT DIRECTION
11140      REMark ANNULEE
11150      REMark ANNULEE
11160      REMark ANNULEE
```



Organigramme 2.3: Suppression des tests inutiles

UN PEU D'ACTION.

Cela va avoir pour effet d'afficher les mots-clés à l'écran, mais il ne se passera rien à proprement parler. Nous allons donc vous présenter 'Boris', la tortue, qui va nous servir de modèle et se déplacer sur l'écran selon nos ordres. Pour plus de commodité et pour bien différencier le texte de l'image, nous allons séparer l'écran en deux fenêtres grâce à la PROCedure ECRAN. Celle-ci coupe ce dernier verticalement avec la fenêtre #1 à gauche (blanche) et la fenêtre #2 à droite (verte). Le texte apparaîtra dans celle de droite,et la trace de 'Boris' dans celle de gauche.

```
12000 DEFine PROCedure ECRAN
12010   MODE 4
12020   WINDOW #1,230,200,257,16
12030   BORDER #1,1,2
12040   CSIZE #1,1,0
12050   PAPER #1,4
12060   INK #1,0
12070   CLS#1
12080   WINDOW #2,230,200,26,16
12090   BORDER #2,1,2
12100   CSIZE #2,1,0
12110   PAPER #2,7
12120   INK #2,0
12130   CLS #2
12140   INK #0,7
12150   CLS #0
12160 END DEFine ECRAN
```

La PROCedure START doit maintenant appeler ECRAN, fixer l'échelle du tracé, et placer la tortue à sa position de départ. Les coordonnées absolues sont 10,10, canal #2, mais il est plus simple d'exprimer les mouvements en déplacements relatifs par rapport à ce point, avec les variables X% et Y%.

```
10010 ECRAN
10020 SCALE #2,20,0,0
10030 LINE #2,10,10
10040 X%=0:Y%=0
```

Le déplacement est géré par la PROCedure TRACE qui dessine une ligne à partir du dernier point (0,0). Remarquez que les modifications sont passées à TRACE par X1 et Y1.

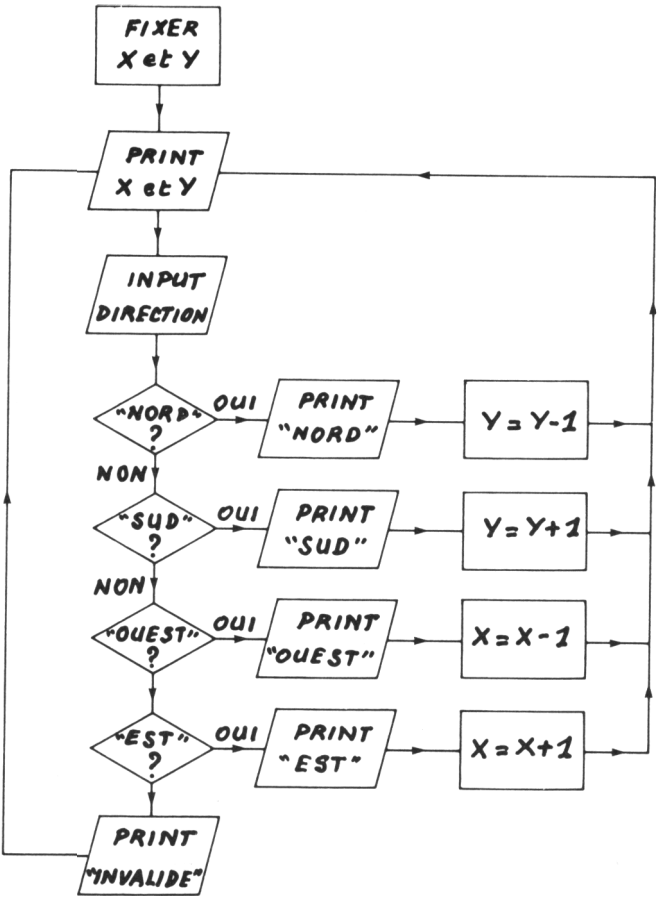
```
13000 DEFine PROCedure TRACE (X1,Y1)
13010   LINE_R #2,0,0 TO X1,Y1
13020 END DEFine  TRACE
```

Il faut maintenant ajouter la réponse à notre ordre, le message indiquant qu'il a été compris, et afficher la position courante. (Organigramme 2.4).

```
11020      UNDER 0:PRINT\"DIRECTION ?"\"X=";X%
,"Y=";Y%
11100       IF CHAINE$="NORD" THEN PRINT "NORD
":Y%=Y%+1:TRACE 0,1:NEXT DIRECTION
11110       IF CHAINE$="SUD" THEN PRINT "SUD":
Y%=Y%-1:TRACE 0,-1:NEXT DIRECTION
11120       IF CHAINE$="OUEST" THEN PRINT "OUE
ST":X%=X%-1:TRACE -1,0:NEXT DIRECTION
11130       IF CHAINE$="EST" THEN PRINT "EST":
X%=X%+1:TRACE 1,0:NEXT DIRECTION
```

UTILISATION DE PROCEDURES DE DIRECTION

Il s'agit ici d'un exemple très simple; cependant lorsque le résultat de vos ordres devient plus complexe, il convient de mettre les réponses dans des PROCedures séparées.



Organigramme 2.4: Réponse.

```
11100      IF CHAINE$="NORD" THEN NORD:NEXT DI
RECTION
11110      IF CHAINE$="SUD" THEN SUD:NEXT DIRE
CTION
11120      IF CHAINE$="OUEST" THEN OUEST:NEXT
DIRECTION
11130      IF CHAINE$="EST" THEN EST:NEXT DIRE
CTION
```

```
14000 DEFine PROCedure NORD
14010    PRINT "VERS LE NORD"
14020    Y%=Y%+1
14030    TRACE 0,1
14090 END DEFine NORD

14100 DEFine PROCedure SUD
14110   PRINT "VERS LE SUD"
14120   Y%=Y%-1
14130   TRACE 0,-1
14140 END DEFine SUD

15000 DEFine PROCedure OUEST
15010   PRINT "VERS L'OUEST"
15020   X%=X%-1
15030   TRACE -1,0
15040 END DEFine OUEST

16000 DEFine PROCedure EST
16010   PRINT "VERS L'EST"
16020   X%=X%+1
16030   TRACE 1,0
16040 END DEFine EST
```

UNE METHODE PLUS SOUPLE.

On peut augmenter le nombre de tests IF-THEN à l'infini, (dans les limites de la mémoire disponible!), mais cette méthode est assez grossière et pose des problèmes lorsque le programme se complique. Une méthode plus souple consiste à mettre mots-clés et réponses en DATA, puis à les mémoriser dans des tableaux alphanumériques.

Il faut d'abord dimensionner correctement C$ pour les mots-clés, et R$ pour les réponses. En SuperBASIC les tableaux alphanumériques doivent être dimensionés à la fois selon le nombre d'éléments (3) et la longueur des chaînes (20) qu'ils doivent contenir. (Avec la dimention 0 on dispose en tout de 4 directions possibles). Il faut ensuite réfléchir à la façon de comparer les chaînes contenues dans le tableau avec celles entrées.

En effet, la longueur d'une chaîne entrée est simplement égale au nombre de caractères qu'elle contient alors que la longueur de celles contenues dans le tableau est toujours de 20, l'espace non utilisé étant "rempli" avec des CHR$ (0).

Si l'on entre NORD, il n'y a pas égalité avec l'élément du tableau qui contient NORD (plus 16 positions vides), sauf si on force le format de la CHAINE$ en lui donnant une dimension DIM (CHAINE$,20).

```
10050  DIM C$(3,20), R$(3,20), CHAINE$(20)
```

Si l'on place les mots-clés et les réponses par paires dans les lignes DATA, il sera plus facile de les associer et de les lire chacun leur tour. (Voir tableau 2.2.)

Numéro d'élément	mots-clés C$	Réponse R$
0	NORD	VERS LE NORD
1	SUD	VERS LE SUD
2	OUEST	VERS L'OUEST
3	EST	VERS L'EST

Tableau 2.2: Contenu des tableaux "mots-clés" et "réponses".

Maintenant nous allons ajouter quelques lignes à la PROCedure START pour initialiser les tableaux (les remplir avec nos mots). Lors du RUN le superBASIC n'exécute pas automatiquement un RESTORE, il faut donc le préciser.

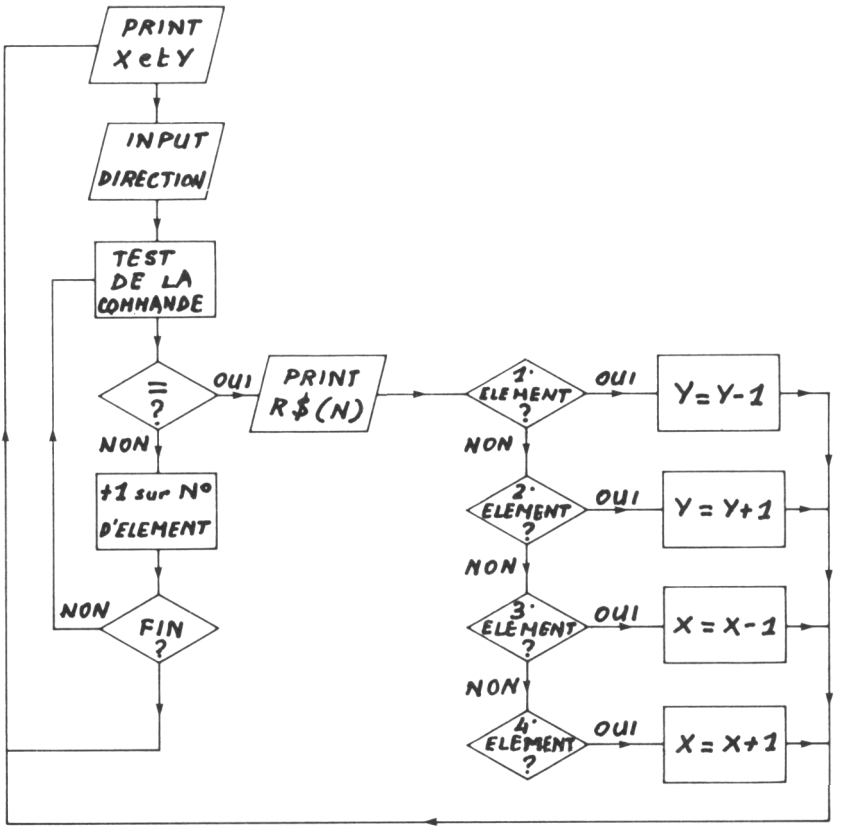
```
10060  RESTORE
10100  DATA "NORD","VERS LE NORD","SUD","VERS
 LE SUD","OUEST","VERS L'OUEST","EST","VERS L
'EST"
10200   FOR N=0 TO 3
10210     READ C$(N):READ R$(N)
10220   END FOR N
```

Tous les tests IF-THEN peuvent être remplacés par une seule boucle qui comparera successivement chaque élément du tableau des mots-clés

(O$) avec le mot que vous aurez entré. (Organigramme 2.5.)

```
11100     FOR N=0 TO 3
11110       IF CHAINE$<>C$(N) THEN
11120     END FOR N
11130     REMark ANNULEE
11140     REMark ANNULEE
11160        ELSE
11170          PRINT R$(N)
11200       END IF
```

A présent, lorsqu'aucun des mots-clés ne correspond avec ceux contenus dans le tableau rien ne se passe; dans le cas contraire (ELSE) la réponse (R$(N)) correspondante vient s'afficher.



Organigramme 2.5: Une méthode plus souple.

Nous nous retrouvons maintenant à l'étape précédente où rien ne se produit vraiment lorsqu'une correspondance est trouvée. Nous allons

donc remédier à ceci en rajoutant de l'action grâce à la PROCedure POSITION qui sera appelée dès qu'un mot-clé sera détecté.

La variable N (qui représente le nombre d'éléments vérifiés dans le tableau) peut être utilisée comme "pointeur" pour indiquer quel mot-clé aura été reconnu. La PROCedure POSITION utilise en effet cette valeur à l'intérieur du test SELECT ON N pour se brancher vers les routines adéquates. Ces dernières sont similaires à celles décrites précédemment si ce n'est qu'il est désormais inutile d'afficher un quelconque message ceci étant fait par R$(N).

```
14000 DEFine PROCedure POSITION
14010   SELect ON N
14020     ON N=0
14030       Y%=Y%+1
14040       TRACE 0,1
14100     ON N=1
14110       Y%=Y%-1
14120       TRACE 0,-1
14140    REMark ANNULEE
14190    REMark ANNULEE
14200     ON N=2
14210       X%=X%-1
14220       TRACE -1,0
14230    REMark ANNULEE
14290    REMark ANNULEE
14300      ON N=3
14310        X%=X%+1
14320        TRACE 1,0
14330   END SELect
14390 END DEFine POSITION
```

EXTENSION DU VOCABULAIRE

Les tableaux peuvent aisément être agrandis pour pouvoir ainsi contenir plus de mots. Si nous définissons une variable WD% qui contient le nombre de mots, nous pourrons l'utiliser successivement pour dimensioner les tableaux ainsi que pour les boucles de lecture et de recherche. Ceci donne une routine générale qui peut être facilement modifiée.

```
10050  WD%=3: DIM C$(WD%,20),R$(WD%,20),CHAIN
E$(20)
10200     FOR N=0 TO WD%
11100      FOR N=0 TO WD%
```

Ajoutons par exemple des directions intermédiaires qui modifient à la fois X et Y.

```
10050   WD%=7: DIM C$(WD%,20),R$(WD%,20),CHAI
NE$(20)
10110   DATA "NORD-EST","VERS LE NORD-EST","S
UD-EST","VERS LE SUD-EST"
10120   DATA "SUD-OUEST","VERS LE SUD-OUEST",
"NORD-OUEST","VERS LE NORD-OUEST"
```

Ajoutons quelques actions :

```
14330   REMark ANNULEE
14390   REMark ANNULEE
14400     ON N=4
14410       Y%=Y%+1:X%=X%+1
14420       TRACE 1,1
14500     ON N=5
14510       Y%=Y%-1:X%=X%+1
14520       TRACE 1,-1
14610     ON N=6
14620       Y%=Y%-1:X%=X%-1
14630       TRACE -1,-1
14700     ON N=7
14710       Y%=Y%+1:X%=X%-1
14720       TRACE -1,1
14730  END SELect
14740 END DEFine POSITION
```

SUPPRESSION DE LA REDONDANCE.

Jusqu'à présent toutes les réponses contiennent l'expression "VERS LE" qui a effectivement été rentrée dans chaque ligne DATA. L'exercice de la frappe peut avoir du bon mais il est tout de même plus logique de remplacer cette expression par une variable alphanumérique une bonne fois pour toute. Remarquez l'espace à la fin de celle-ci pour éviter que le mot qui suit ne soit directement collé. Désormais tous les "VERS LE" peuvent être supprimés des lignes DATA pour être avantageusement remplacés par G$.

```
10100  DATA "NORD","NORD","SUD","SUD","OUEST"
,"OUEST","EST","EST"
10110  DATA "NORD-EST","NORD-EST","SUD-EST","
SUD-EST"
10120  DATA "SUD-OUEST","SUD-OUEST","NORD-EST
","NORD-EST"
10130  G$="VERS LE"
11170  PRINT G$,R$(N)
```

A présent cela apparaît un peu stupide étant donné que les deux tableaux contiennent exactement les mêmes mots; il serait en effet plus logique de supprimer R$ et d'utiliser O$(N) pour afficher aussi les réponses. Il est exact que dans ce cas précis cela ne poserait aucun problème, mais dans le cas où les réponses ne sont pas uniquement une simple répétition des chaînes entrées (ce qui arrive le plus souvent) cela n'est pas possible et un second tableau est indispensable.

Si vous regardez attentivement les PROCedures "d'action",vous vous rendrez compte que leur principal rôle est de mettre à jour les valeurs de X% et Y%. Nous pourrions en fait incorporer ces valeurs dans des lignes DATA et supprimer cette mise à jour du même coup! Il faut, pour cela, créer deux tableaux supplémentaires (pour les coordonnées X et Y), ajouter les valeurs appropriées dans les lignes DATA après chaque réponse et enfin lire (READ) les informations par groupe de quatre (entrée (INPUT), REPONSE, Déplacement en X, Déplacement en Y - voir Tableau 2.3).

```
10050  WD%=7: DIM C$(WD%,20),R$(WD%,20),CHAIN
E$(20),X(WD%),Y(WD%)
10100  DATA "NORD","NORD",0,1,"SUD","SUD",0,-
1,"OUEST",-1,0,"EST","EST",1,0
10110  DATA "NORD-EST","NORD-EST",1,1,"SUD-ES
T","SUD-EST",1,-1
10120  DATA "SUD-OUEST","SUD-OUEST",-1,-1,"NO
RD-OUEST","NORD-OUEST",-1,1
10210   READ C$(N): READ R$(N):READ X(N):READ
 Y(N)
```

On peut maintenant détruire les lignes redondantes et modifier la PROCedure TRACE de

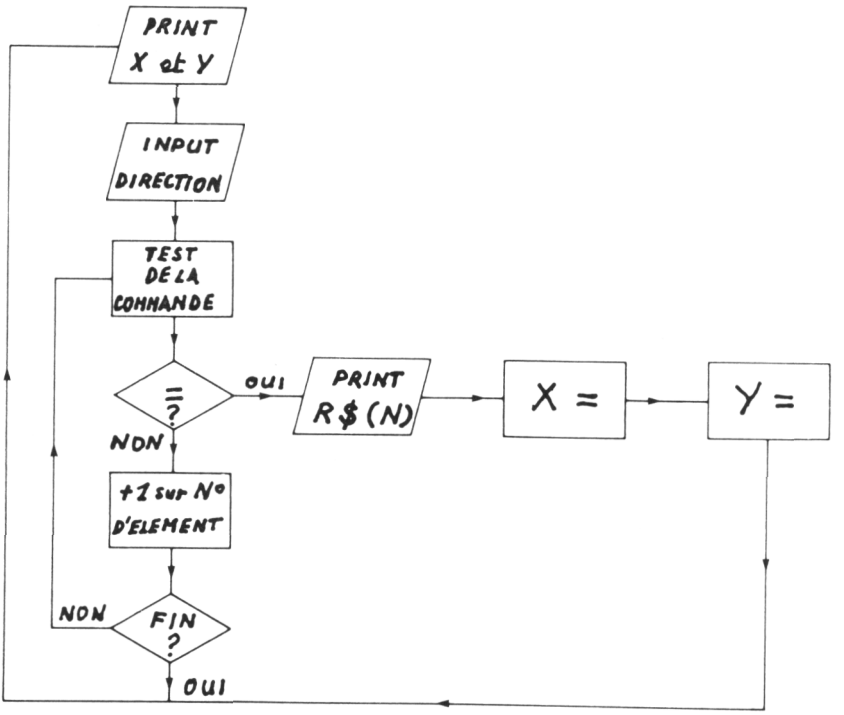
façon à obtenir les bonnes valeurs pour X% et Y% (Organigramme 2.6).

Numéro d'élément	mot-clé C$(N)	Réponse R$(N)	Déplacements X(N)	Y(N)
1	NORD	NORD	0	-1
2	SUD	SUD	0	1
3	OUEST	OUEST	-1	0
4	EST	EST	1	0
5	NORD-EST	NORD-EST	1	-1
6	SUD-EST	SUD-EST	1	1
7	SUD-OUEST	SUD-OUEST	-1	1
8	NORD-OUEST	NORD-OUEST	-1	-1

Tableau 2.3: Déplacements incorporés dans un tableau en X et en Y

```
14010   X%=((X%*3)+(X(N)*3))/3:Y%((Y%*3)+(Y(N
)*3))/3
14020   TRACE X(N),Y(N)
14030   REMark DLINE 14030 TO 14730
```

Cette utilisation d'une série de tableaux liés entre eux est très courante,nous la verrons dans de nombreux programmes.



Organigramme 2.6: Utilisation de tableaux en chaîne.

UTILISATION DE COMMANDES ABREGEES.

Jusqu'ici nous avons donné des ordres en toutes lettres, mais cela exige une frappe longue. Si vous vous sentez un peu paresseux, vous aurez peut-être l'idée de remplacer les ordres par une seule lettre. Cependant, à moins d'utiliser des lettres au hasard, il faudra vérifier que deux ordres ne commancent pas par la même lettre. Pour coder les huit directions ci-dessus nous aurons besoin de deux lettres: N,NE,E,SE,S, SO,O,NO.

```
10100 DATA "N","NORD",0,1,"S","SUD",0,-1
      "O","OUEST",-1,0,"E","EST",1,0
10110 DATA "NE","NORD-EST",1,1,"SE","SUD-EST",1,
      -1


10120   DATA "SO","SUD-OUEST",-1,-1,"NO","NORD-
        OUEST",-1,1
```

Remarquez que seuls les ordres ont changé, les éléments de réponse sont tous dans le second tableau, et restent par conséquent identiques.

EGALITE SUR LA PREMIERE LETTRE

Jusqu'à présent, nous avons toujours vérifié qu'il y avait une correspondance exacte entre la chaîne entrée et les mots-clés. Nous pourrions cependant nous arranger de telle façon que des mots ayant une signification assez proche soient aussi reconnus. Nous pourrions, par exemple, nous contenter de faire un test sur les premières lettres de chaque chaîne uniquement (soit CHAINE$ (1)).

```
11080     CHAINE$=CHAINE$(1)
```

Cela marchera avec NORD, SUD, EST, OUEST, mais il y aura évidemment des problèmes avec les directions intermédiaires: nous les supprimerions une nouvelle fois.

```
10050     WD%=3:DIM C$(WD%,20),R$(WD%,20),CHAI
NE$(20),X(WD%),Y(WD%)
10110     REMark ANNULEE
10120     REMark ANNULEE
```

Par contre il y a beaucoup de mots commençant par N,S,E et O, tous vont être considérés par la machine comme une direction valide.

Par exemple OUI VERS LE NORD donnera VERS L'OUEST.

Une meilleure méthode consiste à comparer quelques lettres et non pas une seule. Dans notre exemple les trois premières lettres des quatre directions principales sont tout à fait caractéristiques.

NOR, SUD, EST, OUE

Si on les utilise comme commandes, les mots suivants seront considérés comme équivalents:

NORD, NORMAL, NORVEGE

Tandis que les mots suivants ne le seront pas:

NAIN, NOM, NATTE.

Nous avons besoin de prendre les trois premières lettres de l'entrée CHAINE$(1 à 3), et de les comparer avec une liste de DATAS modifiée.

```
10100     DATA "NOR","NORD",0,1,"SUD","SUD",0,
-1,"OUE","OUEST",-1,0,"EST","EST",1,0
11080     CHAINE$=CHAINE$(1 TO 3)
```

DES COMMANDES ENCHAINEES.

Dans les routines ci-dessus, nous avons considéré les positions intermédiaires comme des entités particulières, mais si nous pouvions donner plusieurs ordres à la suite, il ne serait pas nécessaire de le faire. Il y a toujours plusieurs façons pour se rendre en un point quelconque, et si plusieurs ordres pouvaient être donnés en une seule fois, il ne serait plus né-

cessaire de se préoccuper des directions comme NORD-EST, qui serait traitée comme une combinaison de NORD et EST.

Ceci nous amène au problème du découpage d'une entrée en plusieurs mots. Comment reconnaître qu'une suite de caractères forment un mot? La réponse la plus évidente est tout simplement qu'il y a des espaces entre eux. Si nous cherchons la présence de blancs dans la chaîne entrée, nous pourrons par conséquent séparer la phrase en plusieurs mots. Le plus simple est d'utiliser la FONction INSTR qui recherche la présence d'une suite de caractères d'une chaîne dans une autre.

Commençons par incorporer INSTR dans une FONction DECOUPAGE, qui est appelée à la ligne 11080.

```
15000 DEFine FuNction DECOUPAGE
15010   SP%=" " INSTR CHAINE$
15030   PRINT "SP% ";SP%
15090 END DEFine DECOUPAGE
```

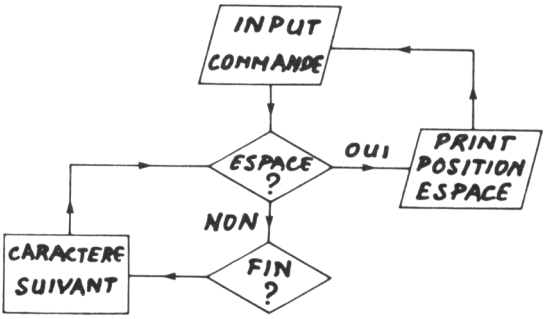
Ici on commence par vérifier si le premier caractère de CHAINE$ est un espace. Si ce n'est pas le cas, le programme continue jusqu'à la fin de CHAINE$. Si aucun espace n'est trouvé, SP% sera mis à 0. Si un espace est trouvé SP% prenprendra comme valeur le nombre de caractères examinés. (Organigramme 2.7) La ligne provisoire 15030 affiche SP% pour observer le fonctionnement de INSTR.

Contrairement à ce que nous avons fait jusqu'à présent, en francais il faudra entrer les directions intermédiaires sans trait d'union, NORD EST et non pas NORD-EST.

Essayons :

NOR OUE

SP%=4



Organigramme 2.7: Localisation des espaces.

NORD OUEST

SP%=5

NOR NOR OUEST

SP%=4

(Remarquons qu'il y aura un message "demande invalide" si CHAINE$ n'est pas traduit par ses trois premiers caractères seulement.)

La longueur du mot est donnée par SP% et seul le premier espace a été trouvé. Pour trouver les autres, il faudra encore faire un effort.

Il faut d'abord ajouter un espace devant le premier mot pour qu'il ait le même format que les autres, puis définir le nouveau point de départ de la recherche (ST%=0).

```
11080      CHAINE$=" " & CHAINE$ : ST%=0
11090      DECOUPAGE
```

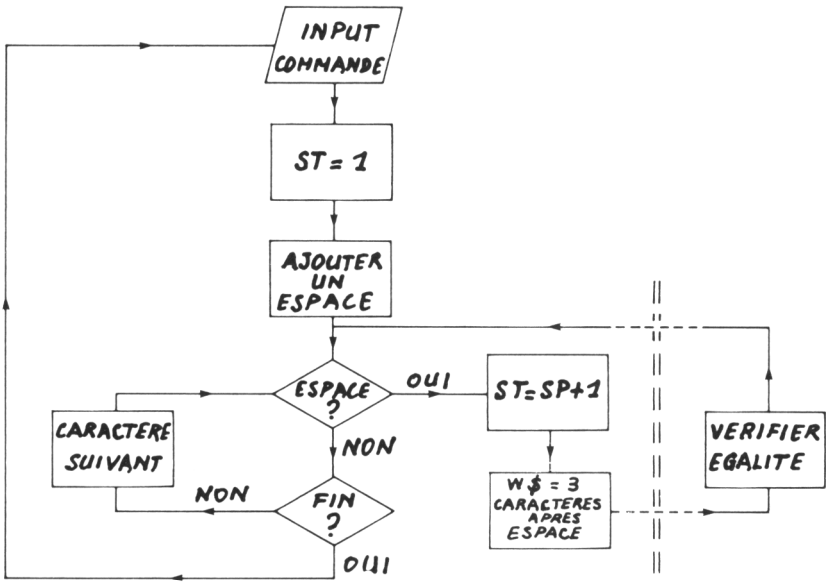
La fonction DECOUPAGE est maintenant modifiée, pour permettre le découpage de toute l'expression entrée. (Organigramme 2.8.) Dès qu'un espace a été trouvé, à SP%, une nouvelle position de départ est fixée un caractère plus loin, à ST%, un mot est découpé trois caractères après l'espace, de ST% à ST%+2, et la CHAINE$ est tronquée de façon à ne contenir que la partie qui

n'a pas été testée, CHAINE$ (ST% TO).

```
15010     SP%=" " INSTR CHAINE$
15030     ST%=SP%+1
15070     W$=CHAINE$(ST% TO ST%+2) : CHAINE$=C
HAINE$(ST% TO)
```

Comme DECOUPAGE a été définie comme FONction, nous pouvons l'utiliser pour renvoyer différentes valeurs. Nous renverrons -1 si aucun espace n'a été trouvé (SP%=0), et 0 s'il y a eu coupure d'un mot.

```
15020     IF SP%=0 THEN RETurn -1
15080   RETurn 0
```



Organigramme 2.8: Découpage d'une expression.

Maintenant utilisons la valeur renvoyée par DECOUPAGE pour sortir de la boucle MOTS dès qu'un mot a été trouvé.

```
11090     IF DECOUPAGE THEN EXIT MOTS
```

Maintenant la frappe de:

NORD OUEST

donne :

VERS LE NORD

VERS L'OUEST

de même :

NOR NOR EST

donne :

VERS LE NORD

VERS LE NORD

VERS l'EST

Il serait beaucoup plus propre de supprimer tous ces "VERS LE" redondants, et de mettre toutes les directions sur une même ligne. Il faut imprimer une seule fois G$, immediatement avant le découpage. Chaque fois que nous comparons le mot courant à ceux qui sont mémorisés, nous imprimons R$(N) s'il y a égalité. Le point-virgule empêchera le saut à la ligne, mais il faudra aussi ajouter des espaces entre les mots.

```
11080      CHAINE$=" " & CHAINE$: ST%=0:PRINT
 G$;
11170         PRINT R$(N);" ";
```

Maintenant:

NORD EST SUD OUEST

donne :

VERS LE NORD EST SUD OUEST.

3

La compréhension du langage naturel

Jusque-là, nous avons dialogué avec l'ordinateur d'une façon restrictive, car il n'est programmé pour comprendre que peu de mots ou de lettres et ne les reconnaît que dans la mesure où ils sont entrés correctement.

Par exemple, si vous mettez un espace avant ou après l'ordre que vous donnez, celui-ci sera rejeté lorsque vous le validerez; cela vient du fait que nous vérifions si les deux chaînes correspondent exactement.

Cependant, dans la vie courante, chacun utilise ce que l'on appelle une "langue naturelle", qui est très élaborée et très variable et que seul le cerveau humain est capable d'appréhender correctement. Même si nous oublions les différences entre le Français et le Canadien ou entre les particularités régionales (est-ce que "je te pique ta voiture" signifie vraiment "je te prends ta voiture"?), une langue pose un nombre infini de problèmes.

Même le système le plus élaboré du monde ne peut répondre à tous les cas: rappelons à ce propos cette vieille histoire: la CIA mit au point un programme de traduction instantané d'Anglais en Russe et réciproquement. Afin d'impressionner le président, ils effectuèrent une démonstration des possibilités de la machine en traduisant tout ce qu'il disait en Russe puis en Anglais. Le président fut très impressionné et très admiratif jusqu'à ce qu'un assistant vienne lui rappeler que Mme la Présidente

l'attendait. Quand il commenta : "loin des yeux, loin du coeur", il fut des plus étonnés d'entendre la machine traduire: "maniaque invisible"!

TRAITER LES PHRASES

Chacun sait que la langue courante est construite avec des phrases. Mais qu'est-ce que l'on entend exactement par phrase? La façon la plus évidente de reconnaître une phrase est qu'elle se termine par un point. Cependant, si nous voulons traiter les phrases, il faut aller beaucoup plus loin.

La définition généralement acceptée d'une phrase est la suivante: "un ensemble de mots, de la langue parlée ou écrite, formant grammaticalement l'expression complète d'une pensée". Elle contient en principe, un sujet et un verbe et définit une constatation, pose une question, donne un ordre ou fait une demande - cette définition précise que la phrase se situe entre deux points de ponctuation. Comment traduire cela en langage courant et a fortiori comment le faire faire à un ordinateur?

DECOMPOSER LA PHRASE

Avant de comprendre une phrase, nous devons la décomposer en ses divers composants afin de pouvoir analyser la signification de chaque segment.

Cette méthode d'analyse grammaticale est bien connue des grammairiens.

Voyons quelques exemples:

J'AIME

Un sujet (J'), un verbe (AIME)

J'AIME LE CHOCOLAT

Un complément d'objet (LE CHOCOLAT)

J'AIME LE CHOCOLAT AMER

Un qualificatif du mot chocolat (AMER)

J'AIME BEAUCOUP LE CHOCOLAT

Un adverbe qui ajoute une qualification au verbe (BEAUCOUP)

Le mot le plus important dans ces propositions est le verbe "AIME" qui traduit l'idée essentielle. La seconde proposition est plus complète car elle précise ce qui est aimé: LE CHOCOLAT. L'adjectif donne une information supplémentaire sur l'objet désiré mais cela devient plus compliqué lorsque l'on ajoute l'adverbe BEAUCOUP.

Voyons maintenant comment un programme d'ordinateur peut décoder une telle phrase. La réponse est de trouver une structure logique de la phrase et les règles que nous pourrions poser pour cet exemple.

1. La phrase commence avec un sujet (J') et finit par un point.

2. Le dernier mot est l'adjectif qualificatif (à moins qu'il n'y ait pas de complément d'objet, et seulement deux mots).

3. Si le mot suivant le verbe n'est pas l'objet, c'est un adverbe; si le mot avant l'objet n'est pas le verbe, c'est un adverbe.

4. La séquence est: groupe sujet-groupe verbe-groupe objet.

Ecrivons un programme dans lequel nous donnons la phrase à l'ordinateur et demandons-lui de la séparer en ses composants. Commençons par définir un format de l'ECRAN avec trois fenêtres. Le canal 0 en bas est réservé à

l'entrée de vos données, le reste de l'écran est divisé en deux fenêtres horizontalement, # 1 et #2. La fenêtre #1 (partie inférieure) montre le résultat final du programme et la fenêtre #2 (partie supérieure) affiche les travaux de certaines sous-routines.

```
30 ECRAN

10000 DEFine PROCedure ECRAN
10010   MODE 4
10020   CLS
10030   WINDOW #0,435,40,36,216
10040   WINDOW #1,455,100,26,116
10050   WINDOW #2,455,100,26,16
10060   BORDER #0,5,4
10070   BORDER #1,3,2
10080   BORDER #2,3,2
10090   PAPER #0,0
10100   PAPER #1,7
10110   PAPER #2,4
10120   INK #0,7
10130   INK #1,0
10140   INK #2,0
10150   CSIZE #0,1,0
10160   CSIZE #1,1,0
10170   CSIZE #2,1,0
10180   CLS #0:CLS#1:CLS#2
```

Nous devons donner une désignation aux compléments d'objets, adjectifs et adverbes pour travailler, en utilisant la PROCedure INIT qui les lit à partir des données d'entrée et les stocke dans les tableaux OB$(n,8), AJ$(n,8) et AV$(n,8) selon leur caractéristique.

Notez que la longueur du mot le plus long (8) doit être prise en compte lorsque l'on dimensionne le tableau et que le nombre de chaque type de mot est défini comme une variable (OB%, AJ%, AV%) afin qu'il soit possible d'en ajouter ultérieurement.

```
10 RESTORE

40 INIT
```

```
11000 DEFine PROCedure INIT
11010   OB%=5: AJ%=5: AV%=2: DIM
        OB$(OB%,10),AJ$(AJ%,10),AV$(AV%,10)
11020   DATA "BISCUIT","CHOCOLATE","GATEAUX",
        "CAFE","THE","EAU"
11030   DATA "FRAIS","AMER","SECS","FROID",
        "BOUILLANT","TIEDE"
11040   DATA "TOUJOURS","BEAUCOUP","SOUVENT"
11050     FOR N=0 TO OB%
11060       READ OB$(N)
11070     END FOR N
11080     FOR N=0 TO AJ%
11090       READ AJ$(N)
11100     END FOR N
11110     FOR N=0 TO AV%
11120       READ AV$(N)
11130     END FOR N
11140 END DEFine INIT
```

Entrons maintenant la phrase à décomposer en utilisant la FuNction SAISIS$(1) décrite auparavant et une boucle de répétition (REPeat CHAINE). N'oubliez pas que vous pouvez utiliser la commande MERGE pour charger la routine SAISIS$ décrite dans le dernier chapitre.

La phrase doit être séparée en mots (voir diagramme 3.1). Pour rendre les choses plus simples, nous ajouterons un espace à la fin de CHAINE$, de telle sorte que le format du dernier mot ressemble à celui des autres, ainsi qu'un caractère fictif juste à la fin pour les raisons données ci-après.

```
120   CHAINE$=CHAINE$ & "*"
```

De nouveau, nous utiliserons l'instruction INSTR pour rechercher les espaces, puis nous séparerons et stockerons chaque mot. Ceci est exécuté ici avec la boucle REPeat MOTS et la fonction STOCKMOT. La recherche initiale est définie par DEB%=1.

```
130   DEB%=1
170     REPeat MOTS
180       IF STOCKMOT THEN EXIT MOTS
190     END REPeat MOTS
```

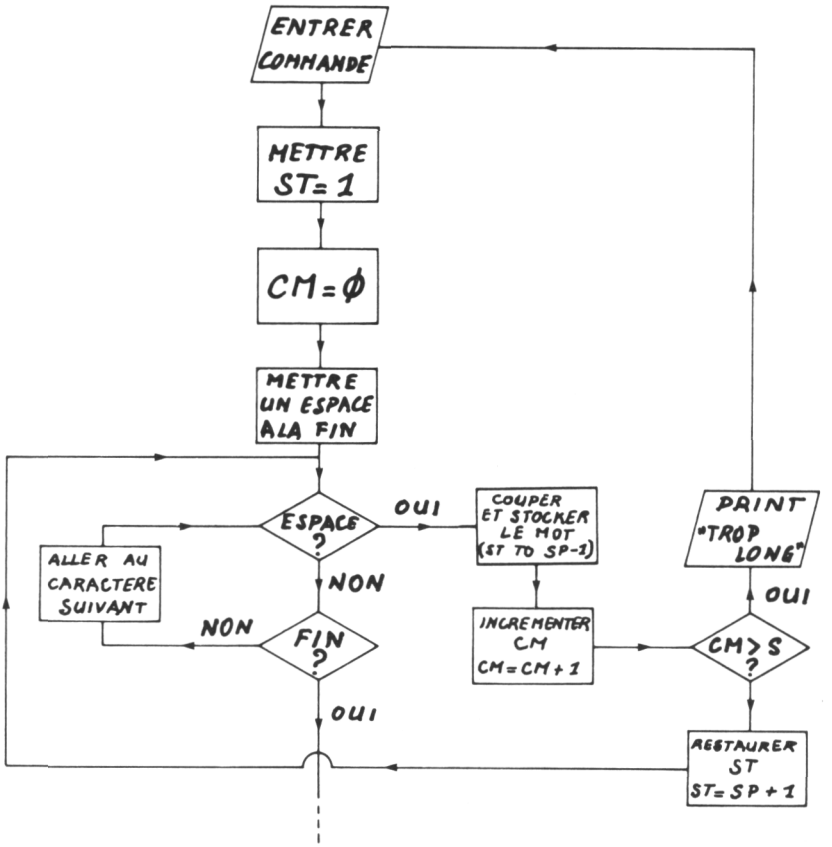
Si un espace n'est pas détecté dans la chaîne (ESP%=0), la fin de la phrase est

atteinte et la fonction STOCKMOT renvoie la valeur -1. Si cette fonction renvoie toute autre valeur que 0, on sort alors de la boucle MOTS (EXIT MOTS).

```
12000 DEFine FuNtion STOCKMOT
12020    ESP%=" " INSTR CHAINE$
12030        IF ESP%=0 THEN RETurn -1
12090 END DEFine STOCKMOT
```

Si un espace est détecté, la partie de CHAINE$ de la position DEB% (début de la recherche) à la position ESP%-1 (espace-1= longueur du mot) est supprimée, le compteur de mots (CM%) est incrémenté, et le mot stocké dans le tableau M$(CM%).

```
20 DIM M$(5,10)
60 CM%=-1
12040    CM%=CM%+1
12060    M$(CM%)=CHAINE$(DEB% TO ESP%-1)
```



Organigramme 3.1: Découpage des mots.

Au début, DEB% est égal à 1, de sorte que la recherche démarre au premier caractère de la chaîne, et que le compteur de mots (CM%) est mis à zéro afin que le premier mot trouvé soit stocké dans la case zéro du tableau stockage des mots.Le compteur de mots est incrémenté à chaque fin de cycle afin que la case suivante du tableau M$ soit utilisée lorsqu'un nouveau mot est détecté.

La longueur de la CHAINE$ est réduite à chaque fois qu'un mot est stocké et celui-ci est supprimé de la chaîne pour ne laisser que CHAINE$(ESP%+1 TO) et la valeur 0 renvoyée par la fonction. Du fait que STOCKMOT est à zéro, la boucle CHAINE est répétée.

L'astérisque à la fin de la chaîne est nécessaire étant donné que la nouvelle CHAINE$ est toujours définie comme un caractère de plus que le dernier espace, de sorte que la fin de CHAINE$ ne doit pas être un espace.

```
12070   CHAINE$=CHAINE$(ESP%+1 TO)
12080   RETurn 0
```

En ajoutant les lignes suivantes, on affichera, dans la fenêtre supérieure, l'opération de réduction de CHAINE$, en cours d'exécution de la recherche.

```
150   UNDER #2,1:PRINT #2,"CHAINE$"\:UNDER #2
,0
12010   PRINT #2,,CHAINE$
```

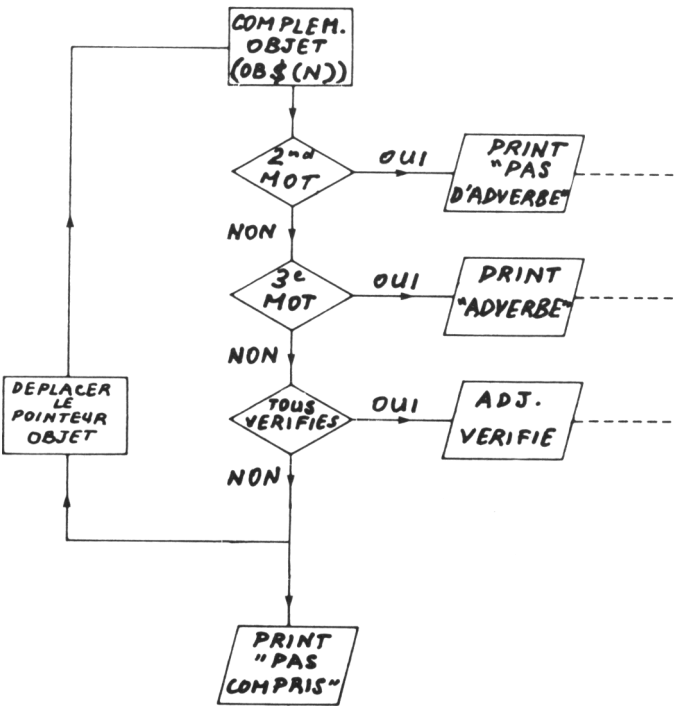
Un contrôle est effectué pour vérifier qu'il n'y a pas plus de 6 mots (0 à 5) dans la phrase, car cela dépasserait les dimensions des tableaux. Si cette condition est vérifiée, CM% est remis à -1, STOCKMOT renvoie la valeur -1 et l'on sort de la boucle (EXIT MOTS).

```
150       UNDER #2,1:PRINT #2,"CHAINE$"\:UNDER
 #2,0
12010       PRINT #2,, CHAINE$
```

Lorsque la recherche est finie (END REPeat MOTS), la liste des mots trouvés est affichée dans la fenêtre du bas.

```
160     UNDER #1,1 : PRINT #1, "M$(N)"\:UNDER
 #1,0
200       FOR N=0 TO CM%
210         PRINT, M$(N);" "
220       END FOR N
```

Un test est alors fait pour vérifier s'il existe une correspondance entre les mots de la phrase M$(N) (voir diagramme 3.2). Seuls les mots 2 et 3 sont vérifiés car ce sont les seules positions possibles des compléments d'objet dans notre phrase type. On se branche sur 2 PROCédures différentes selon la position du mot correspondant dans la phrase. S'il n'y a pas de correspondance, un message est alors affiché et une nouvelle entrée demandée.



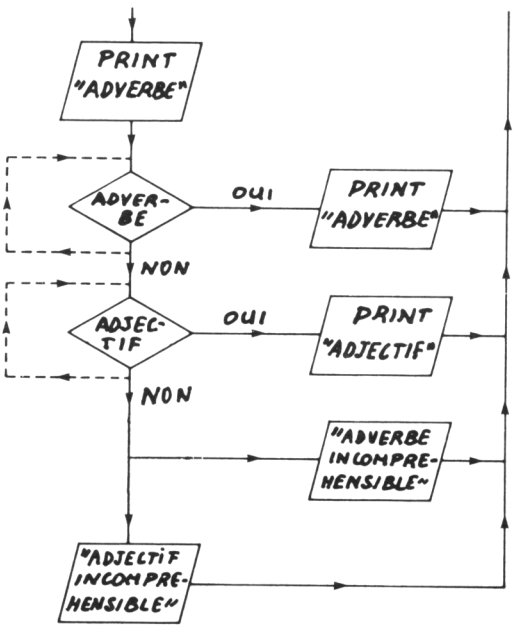
Organigramme 3.2: Chercher une correspondance.

```
230            FOR N=0 TO OB%
240                 IF M$(2)=OB$(N) THEN PAS_D
_ADV:NEXT CHAINE
250                 IF M$(3)=OB$(N) THEN ADV:N
EXT CHAINE
260            END FOR N
270        PRINT \,"Complément d'objet non tr
ouvé"
280  END REPeat CHAINE
```

Si le complément d'objet est détecté comme mot 3, c'est qu'il n'y a pas d'adverbe (mais peut-être un adjectif).

```
1000  DEFine PROCedure PAS_D_ADV
1010    PRINT\,"pas d'adverbe"
1020    ADJ_VERIF 3
1030  END DEFine PAS_D_ADV
```

Si le complément d'objet est détecté comme mot 4 c'est qu'il y a un adverbe et peut-être un adjectif. L'adverbe peut se trouver en première ou en deuxième position (voir diagramme 3.3).



Organigramme 3.3: Adverbe et peut-être adjectif.

```
2000 DEFine PROCedure ADV
2010   FOR n=0 TO AV%
```

Nous vérifions d'abord la position de l'adverbe.

```
2020      IF M$(0)=AV$(n) THEN PRINT,"adverbe
en premiere position": ADJ_VERIF 4:RETurn
2030      IF M$(1)=AV$(n) THEN PRINT,"adverbe
en seconde position":ADJ_VERIF 4:RETurn
2040  END FOR n
```

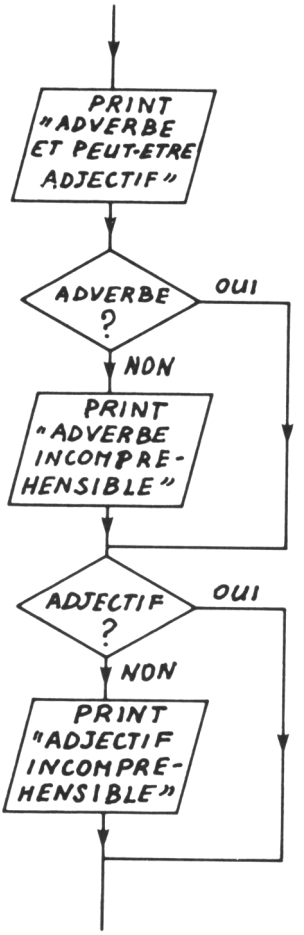
Dans ces deux cas nous vérifions systématiquement si après le complément d'objet il y a un adjectif (ADJ_VERIF).

Notez que la place de l'adjectif se trouve en position 5 (M$(4)); en revanche lorsqu'il n'y a pas d'adverbe l'adjectif se trouve en position 4 (M$(3)). La variable "pos" qui est passée lors de l'appel de la PROCedure ADJ_VERIF assure que c'est bien le bon mot qui est vérifié.

```
3000 DEFine PROCedure ADJ_VERIF (pos)
3010   FOR N=0 TO AJ%
3020      IF M$(pos)=AJ$(N) THEN PRINT "ADJEC
IF":RETurn
3030   END FOR N
3040      IF M$(pos)<>"" THEN PRINT "adjectif
incomprehensible": RETurn
3050      PRINT\,"pas d'adjectif"
3060 END DEFine ADJ_VERIF
```

En dernier lieu, si l'adverbe n'a pas été trouvé, nous l'indiquons.

```
2050      PRINT\,"ADVERBE INCOMPREHENSIBLE":RE
Turn
2060 END DEFine ADVERBE
```



Organigramme 3.4: Adverbe et Adjectif.

LA PONCTUATION

Ainsi que nous l'avons précédemment dit, la fin d'une phrase se reconnaît à son point final, bien qu'habituellement, lorsque l'on rentre une phrase dans la machine, on ne s'en préoccupe pas. Mais qu'arrive-t-il si un utilisateur pointilleux emploie une ponctuation correcte? Si vous réfléchissez à cette question, vous réalisez que l'ordinateur va buter sur une difficulté car il ne reconnaîtra pas le dernier mot, puisque celui-ci sera découpé comme le mot plus un point final.

Nous avons donc besoin de vérifier si le dernier caractère de la chaîne entrée (CHAINE$) est un point final. Le meilleur moment pour vérifier la PONCTUATION semble être juste après l'entrée de la phrase. Si le dernier caractère (FIN$=CHAINE$(LEN(CHAINE$))) est un point final, il suffit simplement de la supprimer et de revenir au traitement habituel.

```
90 PONCTUATION
6000 DEFine PROCedure PONCTUATION
6010   FIN$=CHAINE$(LEN(CHAINE$))
6020       IF FIN$="." THEN SUPPRIME:RETurn
6050 END DEFine PONCTUATION
7000 DEFine PROCedure SUPPRIME
7010   CHAINE$=CHAINE$(1 TO LEN(CHAINE$)-1)
7020 END DEFine SUPPRIME
```

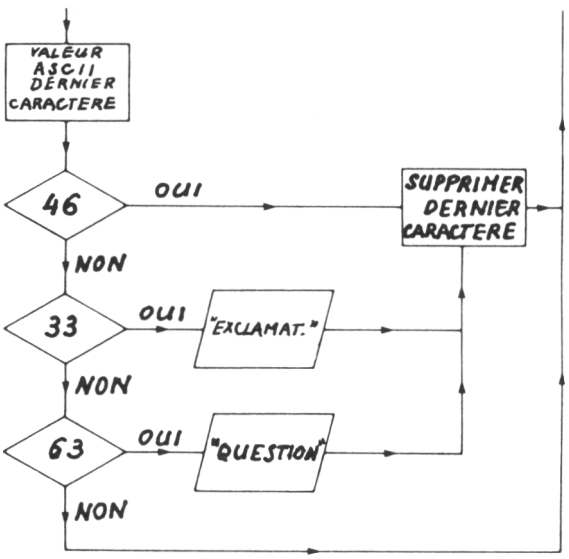
D'autres types de ponctuation peuvent terminer une phrase, il faut donc voir de plus près le dernier caractère. Les plus connus sont les points d'exclamation et d'interrogation qui indiquent le contexte de la phrase.

```
6030       IF FIN$="?" THEN SUPPRIME : PRINT #
0,"QUESTION ?"\
6040       IF FIN$="!" THEN SUPPRIME : PRINT #
0,"EXCLAMATION"\
```

Dans de nombreux BASICs, la commande INPUT n'accepte rien après la virgule, qui est lue comme fin de donnée. Heureusement, il n'en est rien avec le superBASIC :les virgules sont utiles comme indicateurs des différentes parties d'une phrase que l'on veut considérer comme des éléments intrinsèques. Cependant, dans les cas simples, il faut mieux les enlever et les remplacer par des espaces avant que la phrase ne soit séparée en mots. Remarquons que cela fonctionnera d'une façon tout à fait correcte s'il n'y a pas d'espace après la virgule, car tout espace suivant une virgule remplacée sera considéré comme un nouveau mot. Si aucune virgule n'est détectée (VI%=0), on sortira de la boucle; dans le cas contraire, la partie gauche

de CHAINE$ (jusqu'à la virgule) et la partie droite de CHAINE$ (au-delà de la virgule) sont prises et réunies avec un espace entre elles.

```
8000 DEFine PROCedure VIRGULE
8010   REPeat VIRBOUCLE
8020     VI%=",," INSTR CHAINE$
8030       IF VI%=0 THEN RETurn
8040     CHAINE$=CHAINE$(1 TO VI%-1) & " " &
         CHAINES$(VI%+1 TO)
8050   END REPeat VIRBOUCLE
8060 END DEFine VIRGULE
```



Organigramme 3.5: Ponctuation.

Les apostrophes peuvent être traitées de la même façon, si ce n'est que nous ne les remplacerons pas par un espace mais que nous les supprimerons purement et simplement.

```
110     APOSTROPHE
9000 DEFine PROCedure APOSTROPHE
9010   REPeat APOSBOUCLE
9020     AP%="'" INSTR CHAINE$
9030       IF AP%=0 THEN RETurn
9040     CHAINE$=CHAINE$(1 TO AP%-1) &
         CHAINE$(AP%+1 TO)
9050   END REPeat APOSBOUCLE
9060 END DEFine APOSTROPHE
```

UNE METHODE DE RECHERCHE ITERATIVE

Bien que la méthode que nous avons décrite précédemment fonctionne correctement, elle a l'inconvénient d'exiger d'entrer une phrase d'un nombre limité de mots.

Utiliser une méthode itérative sur la phrase entière - pour chaque mot-clé - sans diviser au préalable la phrase de n'importe quel format. Dans cette méthode, nous prenons le premier mot significatif et essayons de le comparer au même nombre de lettres que dans CHAINE$, en partant du premier caractère. Si ce test ne marche pas, il est automatiquement répété, cette fois-ci à partir du second caractère, et ainsi de suite, jusqu'à ce qu'une correspondance soit trouvée ou bien que l'on atteigne la fin de la CHAINE$. Par exemple si CHAINE$ est "J'AIME LE CAFE" et que le premier mot significatif soit CAFE, nous aurons:

```
J'AI
'AIM
AIME
IME
ME L
E LE
LE C
E CA
 CAF
CAFE (correspondance trouvée)
```

Nous pouvons utiliser une grande partie de notre programme mais de nombreux changements doivent être faits. Il faut supprimer les lignes 80 à 9999 - avec la commande "DLINE 80 TO 9999" et modifier l'instruction DIM de la ligne 20, de façon à porter la capacité du tableau M$ à 20. La fonction STOCKMOT n'est pas utilisée non plus, vous pouvez par conséquent la supprimer "DLINE 12000 TO 12090".

```
10 RESTORE
20 DIM M$(19,10)
30 ECRAN
```

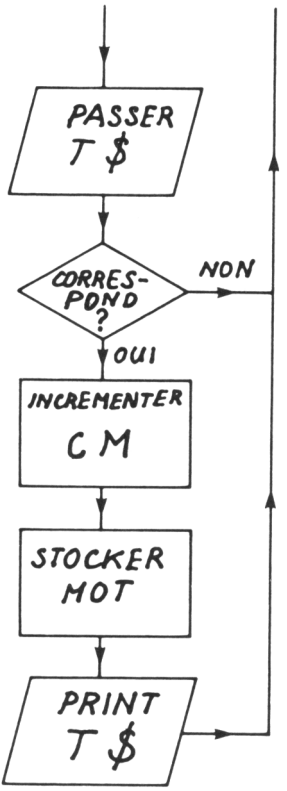
```
40 INIT
50    REPeat CHAINE
60      CM%=-1
70      AT #0,1,1: CHAINE$=SAISIS$(1)
90      CLS#1 : CLS#2
```

A la place de la fonction STOCKMOT, nous utiliserons une procédure à peu près similaire: TROUVE(T$) (voir le diagramme 3.6). T$(paramètre de la procédure TROUVE) contient temporairement le mot qui sera recherché dans CHAINE$. T$ peut donc être utilisé pour rechercher, grâce à la fonction INSTR, une quelconque correspondance. Si la recherche ne donne pas de résultat, la procédure est abandonnée et l'on revient à l'endroit où elle a été appelée (RETurn). Pour se rendre compte de ce qui a été trouvé, et pour pouvoir utiliser ultérieurement les mots détectés, nous les stockerons immédiatement. Nous venons d'augmenter la capacité du tableau M$, il n'y a par conséquent pas de problème (même pour les phrases les plus longues!).

```
1000 DEFine PROCedure TROUVE(T$)
1010   CHAINE%=T$ INSTR CHAINE$
1020      IF CHAINE%=0 THEN RETurn
1040   CM%=CM%+1
1050   M$(CM%)=T$
1080   PRINT #2,,,T$
1090 END DEFine TROUVE
```

Tous les compléments d'objet peuvent être détectés dans CHAINE$ grâce à une boucle, et il en est de même pour les adverbes et les adjectifs.

```
120   FOR M=0 TO OB%
130     TROUVE OB$(M)
140   END FOR M
150   FOR M=0 TO AV%
160     TROUVE AV$(M)
170   END FOR M
180   FOR M=0 TO AJ%
190     TROUVE AJ$(M)
200   END FOR M
```



Organigramme 3.6: Trouve (T$).

L'exécution du programme s'arrête alors jusqu'à ce que le temps de pause soit atteint (500), ou qu'une touche soit enfoncée, avant que la fenêtre où figure la phrase entrée soit effacée pour permettre l'entrée d'une autre phrase.

```
250       FICTIF$=INKEY$(500)
260       CLS #0
270   END REPeat CHAINE
```

CORRESPONDANCE PARTIELLE

L'avantage de la méthode itérative est que l'on peut arriver à obtenir une reconnaissance d'une série de mots liés en analysant seulement quelques caractères clés. Cela est évidemment

très utile, car il n'est alors plus nécessaire de faire la distinction entre les pluriels et les singuliers. Si vous corrigez la ligne 11020 comme ci-dessous:

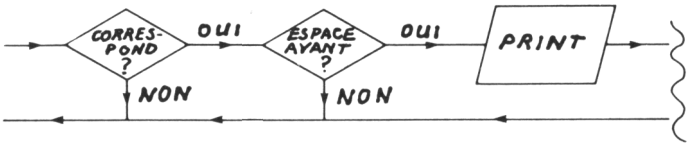
```
11020 DATA "BISCUIT","CHOCOLAT","GATEAU","CAFE",
      "THE","EAU"
```

Les deux mots "GATEAUX" et "GATEAU" seront reconnus.

Cependant les choses ne sont pas aussi simples, car utiliser THE plutôt que THES peut conduire à des résultats inattendus. On pourra détecter THE et THES, mais aussi THEIERE, THEATRE, THEISME, THEME de la même façon.

Ce problème n'est pas limité aux seuls préfixes, la machine ne fera pas non plus de différence entre AMER et RAMER. Vous pourriez intégrer une routine qui vérifierait que le caractère précédent le mot à détecter est bien un espace, ce qui éviterait ce genre de problème (voir diagramme 3.7). CHAINE$ donne la position du début du mot, de sorte que CHAINE$(CHAINE%-1) donne le caractère précédent qui provoque le retour à la procédure d'appel si le caractère est bien un espace.

```
1030      IF CHAINE$(CHAINE$-1)<>" " THEN RETu
rn
```



Organigramme 3.7: Vérification du début d'un mot.

Pour que cette routine fonctionne correctement sur le premier mot de la chaîne, il faut insérer un espace au début de CHAINE$.

```
80  CHAINE$=" " & CHAINE$
```

De la même façon, vous pourriez concevoir une routine similaire pour vérifier le caractère qui suit le mot détecté, ou bien la longueur de celui-ci, afin de restreindre les erreurs

possibles.

METTRE LES CHOSES EN ORDRE

Bien que nous ayons détecté tous les mots de la phrase, sans prêter attention, ni à leur position, ni à leur nature, ils sont placés dans l'ordre dans lequel ils apparaissent dans la ligne DATA. Cela parce que la comparaison commence par le premier mot de la phrase. Il serait utile de pouvoir réaménager le tableau dans lequel ils apparaissent dans la phrase.

Pour cela, nous devons garder la trace de la position du mot dans la phrase, CHAINE% et le compteur de mots (CM%), car chaque mot est comparé à un nouveau tableau (PM%) (Position des Mots). C'est un tableau à deux dimensions où la position dans la phrase est contenue dans le premier élément, PM%(CM%,0), et le compteur de mots, PM%(CM%,1) dans le second. Pour rendre l'affichage plus clair, les termes "mot" et "position" (c'est-à-dire la position de départ du mot détecté dans la phrase) ont été rajoutés.

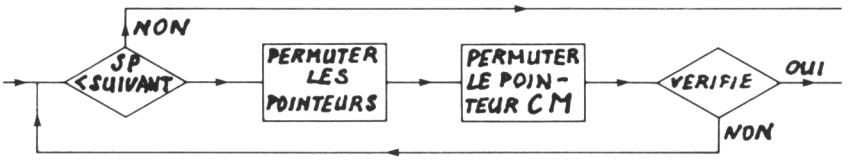
```
20    DIM M$(19,10) : DIM PM(19,1)
110       PRINT ,"mot","position"
1060     PM(CM%,0)=CHAINE% : PM(CM%,1)=CM%
1070     PRINT #2,,PM(CM%,1),PM(CM%,0);
```

La routine qui effectue vraiment l'ordonnancement (PROCedure ORDRE) n'est appelée que si une correspondance est trouvée.

```
210    ORDRE
```

La boucle de TRI agit sur un simple échange (voir organigramme 3.8). Elle extrait la position du premier mot trouvé dans la phrase (IS%) (premier élément dans la première dimension, PM(0,0)) et la compare avec la position du second mot détecté (deuxième élément dans la première dimension PM(0+1,0)). Si la première position est plus grande que la seconde cela signifie que le premier mot détecté dans la

phrase est plus loin que le second et que, par conséquent, il doit être interverti grâce à la variable D%. Cela aura pour effet de remettre les pointeurs de position en ordre, mais il ne faut pas oublier que le compteur de mots doit aussi être réordonné. Cette méthode est répétée jusqu'à ce que tous les détecteurs de mots soient dans un ordre correct. Notez que le véritable contenu du tableau qui contient les mots n'est pas modifié; seuls les pointeurs "index" le sont.



Organigramme 3.8: Mettre les mots dans l'ordre.

```
100      PRINT #2,,"stock","position"
2000 DEFine PROCedure ORDRE
2010   REPeat TRI
2020     FOR n=0 TO CM%-1
2030       IF PM(n=0)>=PM(PM(n+1,0) THEN
2040         D%=PM(n,0) : PM(n,0)=PM(n+1,0)
             : PM(n+1,0)=D%
2050         D%=PM(n,1) : PM(n,1)=PM(n+1,1)
             : PM(n+1,1)=D%


2060   END REPeat TRI
2070       END IF
2080     END FOR n
2090 END DEFine ORDRE
```

Si les chaînes sont maintenant imprimées dans l'ordre revu du compteur de mots, CM%, elles apparaîtront comme elles étaient dans la phrase d'origine, ce qui facilite la compréhension.

```
220      FOR n=0 TO CM%
230        PRINT , n+1,PM(n,0),,M$(PM(n,1))
240      END FOR n
```

Si vous exécutez ce programme avec plusieurs phrases différentes, vous pourrez visualiser la

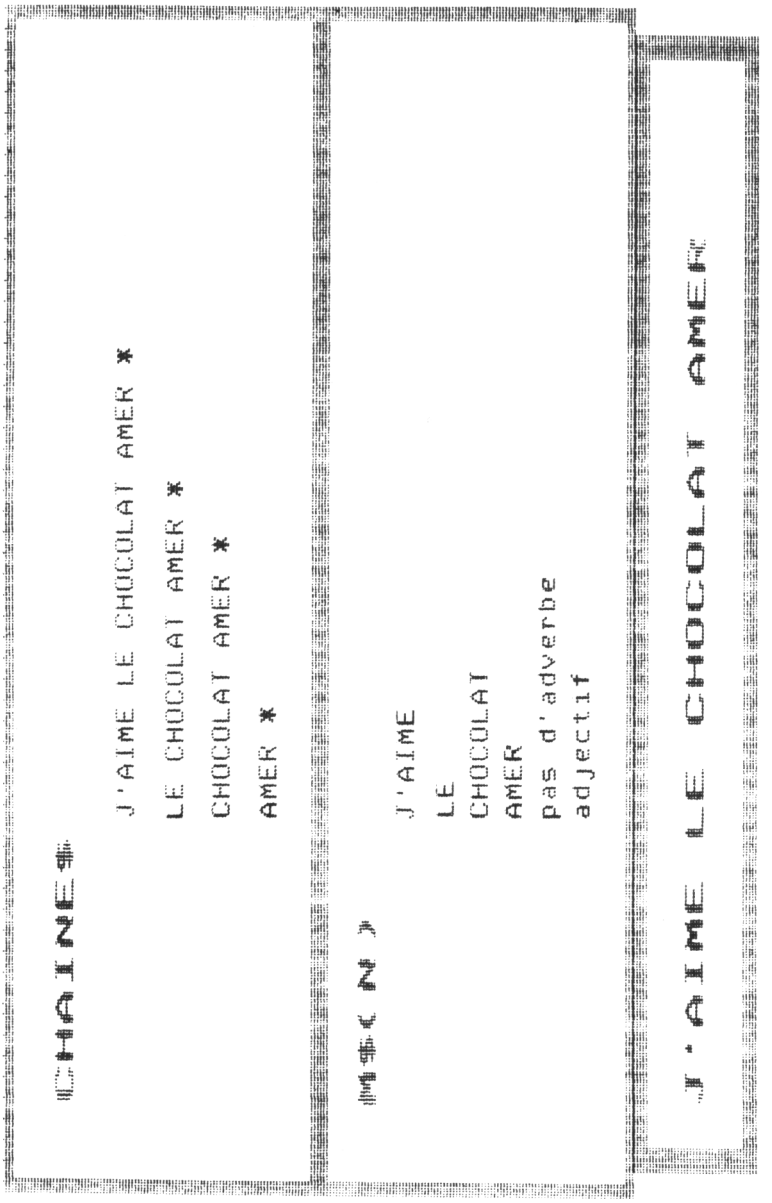

CHAINE#
J'AIME LE CHOCOLAT AMER *
LE CHOCOLAT AMER *
CHOCOLAT AMER *
AMER *
MEC N)
J'AIME
LE
CHOCOLAT
AMER
pas d'adverbe
adjectif
J'AIME LE CHOCOLAT AMER

Figure 1: Tri.

position originale des mots dans la fenêtre supérieure, et les mots réordonnés comme dans la phrase dans la fenêtre du bas.

4

La construction des réponses

DES REPONSES SENSEES

Jusqu'ici nous avons vu comment décoder une phrase introduite en machine, mais les réponses obtenues étaient très formelles et très limitées. Bien que les mêmes mots soient souvent utilisés dans les questions et les réponses, dans la conversation normale nous analysons le sujet de la phrase et le modifions en fonction du contexte de la réponse.

Par exemple:

J'AIME LES CHOCOLATS

pourrait entraîner le commentaire suivant:

IL AIME LES CHOCOLATS

et, de la même façon:

IL AIME LES CHOCOLATS

devrait donner:

J'AIME LES CHOCOLATS

On constate dans ces exemples, que pour chaque entrée d'un sujet, on trouve un sujet équivalent dans le commentaire, et qu'il suffit d'inverser les sujets et d'ajouter le reste de la phrase pour obtenir le résultat souhaité.

"J'" contient deux signes, ainsi nous pouvons vérifier CHAINE$(1 TO 2), et le remplacer par "IL" et ajouter le reste de la phrase CHAINE$(3 TO).

```
10 ECRAN
20    REPeat BOUCLE
30      AT#0,1,1:CHAINE$=SAISIS$(1)
40      CHAINE$=CHAINE$ & " "
60       IF CHAINE$(1 TO 2)="J'" THEN PRINT "
IL " & CHAINE$(3 TO )
90     FICTIF$=INKEY$(500)
100    CLS#0
110   END REPeat BOUCLE
```

(Notez que le format de l'ECRAN et la routine SAISIS$(1) sont exactement les mêmes que dans les programmes précédents.)

De la même façon, les deux premiers caractères pourraient être remplacés par "J'" au cas où "IL" serait introduit.

```
80       IF CHAINE$(1 TO 2)="IL" THEN PRINT "J
'" & CHAINE$(3 TO )
```

Vous constaterez que cela fonctionne parfaitement jusqu'à ce que vous tapiez quelque chose comme:

ILS AIMENT LES CHOCOLATS

qui donnera:

J'SAIMENT LES CHOCOLATS

Qui ne donne pas vraiment un résultat satisfaisant!

Nous pourrions traiter ce problème en vérifiant globalement "J'AI" et "IL A", comme on contrôle "J'" et "IL", mais pour cela il faut modifier les lignes 60 et 80, pour éviter les interférences.

```
60      IF CHAINE$(1 TO 4)="J'AI" THEN PRINT "
IL A" & CHAINE$(5 TO )

80     IF CHAINE$(1 TO 4)="IL A" THEN PRINT "J
'AI" & CHAINE$(5 TO)
```

UNE AUTRE DIMENSION

Bien que cette méthode fonctionne, rapidement on va se trouver confronté à des problèmes de place considérable, car il faut une ligne pour chaque hypothèse étant donné que nous devons prendre en compte la longueur du mot ou de la phrase correspondante. Lorsque plusieurs mots doivent être contrôlés, il est préférable d'utiliser un tableau à plusieurs dimensions i$(N,M) qui - au sein d'une boucle - va pouvoir servir d'élément de référence avec la phrase entrée.

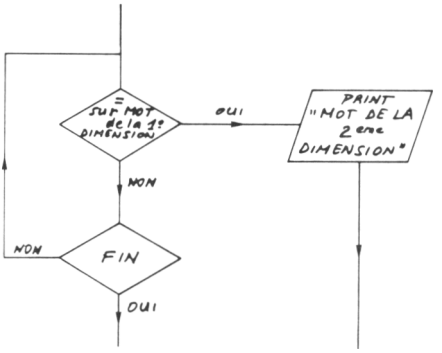
La solution la plus simple consiste à ranger les éléments à comparer avec la phrase entrée dans la première dimension, i$(N,0), du tableau et ceux de la réponse dans la seconde i$(N,1). Il est plus facile d'éviter les erreurs en entrant les données, grâce à l'instruction DATA, qui seront lues dans l'ordre au sein de la PROCédure INIT.

Soit la PROCedure:

```
20 INIT


11000 DEFine PROCedure INIT
11010   RESTORE
11030   DIM i$(1,1,4)
11100   DATA "J'AI","IL A","IL A","J'AI"
11200    FOR n=0 TO 1
11210      READ i$(n,0) : READ i$(n,1)
11220    END FOR n
11390 END DEFine INIT
```

Une nouvelle fois, nous utiliserons une boucle de recherche itérative, qui pour le moment imprimera simplement le mot ou la phrase

correspondant à ceux reconnus i$(N,1) (voir organigramme 4.1). Un des avantages de cette méthode, est qu'elle permet de reconnaître n'importe quelle correspondance de mots, qu'ils soient entourés d'espaces ou non; en effet, nous ne séparons pas la phrase en mots comme nous l'avons fait dans les précédents chapitres. (Notez une fois encore, que les procédures ECRAN et SAISIS$(1) sont les mêmes qu'auparavant.)



Organigramme 4.1: Une réponse correspondant.

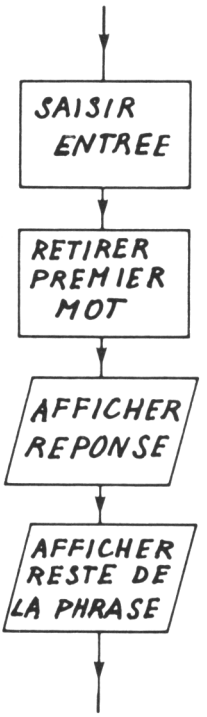
```
10 ECRAN
30  REPeat CHAINE
40    AT#0,1,1 : CHAINE$=SAISIS$(1)
50    CHAINE$=CHAINE$ & " "
60      FOR M=0 TO 1
70       IS%=i$(M,0) INSTR CHAINE$
80          IF IS%>0 THEN EXIT M
90      END FOR M
100    PRINT#2, i$(M,1),CHAINE$(IS%+LEN(i$(M,
0)) TO )
110    PRINT i$(M,1) & CHAINE$(IS%+LEN(i$(M,0
)) TO )
120   FICTIF$=INKEY$(500)
130   CLS#0
140  END REPeat CHAINE
```

La réponse désirée se trouve dans la seconde dimension du tableau i$(M,1), de sorte que nous donnons l'ordre de l'afficher lorsque l'on quitte la boucle.

```
100    PRINT#2, i$(M,1)
```

Pour obtenir une réponse plus complète, il est nécessaire de rajouter le reste de la phrase originale (voir organigramme 4.2). Il n'y a aucune difficulté à définir "le reste de la phrase" étant donné qu'il suffit de soustraire le mot détecté du début de la phrase. IS% marque le début de la correspondance, et nous pouvons très facilement déterminer la longueur du mot trouvé étant donné qu'il se trouve dans la première dimension du tableau i$(M,0). Par conséquent, il faut rajouter CHAINE$(IS%+LEN(i$(M,0))) à la réponse. Pour faire apparaître clairement ce qui se produit, chaque partie de la réponse vient s'afficher dans la fenêtre du haut.

```
100     PRINT#2, i$(M,1),CHAINE$(IS%+LEN(i$(M,
0)) TO )
110     PRINT i$(M,1) & CHAINE$(IS%+LEN(i$(M,0
)) TO )
```



Organigramme 4.2: Une réponse plus complète.

Maintenant si vous essayez:

J'AI ENVIE DE MANGER

L'ordinateur confirme:

IL A ENVIE DE MANGER

DES REPONSES PLUS ELABOREES

Nous avons donc résolu le problème de la reconnaissance des groupes sujets pour pouvoir les exploiter dans un traitement simple d'une phrase. Rappelez-vous que vous pouvez rajouter autant de sujets que vous le désirez, à condition d'augmenter la dimension du tableau i$(M,N) ainsi que la variable MC% (qui contient le nombre de mots dans les lignes DATA). Bien que cette solution présente un avantage non négligeable par rapport aux solutions précédentes, il n'en reste pas moins qu'il faut introduire tous les verbes à toutes les personnes si nous voulons un traitement plus complet des phrases.

Dans la langue française la plupart des verbes se conjuguent à partir d'une racine suivie d'une terminaison qui est toujours la même selon la personne et le groupe auquel appartient le verbe. Ainsi la solution la plus intéressante, si l'on veut aller plus loin, est de continuer à stocker les sujets comme auparavant (c'est-à-dire dans un tableau à deux dimensions S$(N,0) et S$(N,1), l'une contenant les mots à reconnaître et l'autre les sujets pour la réponse), puis de stocker dans un autre tableau V$(N) les racines des verbes qui sont communes à toutes les personnes (nous avons sélectionné ici quelques verbes du premier groupe). Enfin dans deux autres tableaux RE$(N), nous stockons des phrases préparées, destinées à répondre aux phrases qui sont entrées, puis dans le tableau TE$(N) les terminaisons correspondant à chaque personne.

Nous avons donc la PROCédure d'initialisation suivante:

```
10 ECRAN
20 INIT


11000 DEFine PROCedure INIT
11010   RESTORE
11020   SU%=7:VB%=6:RP%=6
11030   DIM S$(SU%,1,7) : DIM V$(VB%,7) : DIM
 RE$(RP%,50): DIM TE$(SU%,1,5)
```

Les premières lignes de DATA contiennent les sujets - à la fois pour la reconnaissance et pour la réponse - ainsi qu'ils sont disposés ci-dessous:

S$(n,0)	S$(n,1)
JE	TU
TU	JE
IL	ELLE
ELLE	IL
NOUS	VOUS
VOUS	NOUS
ILS	ELLES
ELLES	ILS

Bien évidemment l'ordre de ces sujets peut être modifié selon l'orientation que l'on souhaite donner aux réponses.

```
11040  DATA "JE ","TU ","TU ","JE ","IL ","EL
LE"
11050  DATA "ELLE","IL","NOUS ","VOUS ","VOUS
 ","NOUS "
11060  DATA "ILS ","ELLES ","ELLES","ILS"


11140     FOR n=0 TO SU%
11150       READ S$(n,0):READ S$(n,1)
11160     END FOR n
```

Les lignes DATA suivantes contiennent les racines des verbes; comme nous l'avons précédemment dit seuls quelques verbes du premier groupe

ont été sélectionnés, nous avons volontairement écarté les verbes irréguliers, beaucoup plus difficiles à traiter.

```
11100  DATA "DETEST","CONSTAT","RAFFOL","REV"
,"SOUHAIT","CHANT","JOU"
```

```
11170      FOR n=0 TO VB%
11180        READ V$(n)
11190      END FOR n
```

Les lignes suivantes contiennent les réponses toutes faites selon les verbes qui seront employés dans la phrase entrée. Ces phrases peuvent bien sûr être modifiées à volonté si l'on désire changer le sens des réponses. Notez qu'à chaque fois c'est le verbe qui apparaît dans la phrase entrée qui est réutilisé dans la réponse.

```
11110  DATA " CA PROBABLEMENT AUTANT"," DES P
ROGRES FULGURANTS"
11120  DATA " D'UN TAS DE CHOSES"," D'UN JOUR
 MEILLEUR"
11130  DATA " , MAIS RIEN NE DIT QUE CA SE RE
ALISERA"," PARFAITEMENT JUSTE"," BEAUCOUP MIE
UX QU'ON NE POURRAIT LE CROIRE"
```

```
11200      FOR n=0 TO RP%
11210        READ RE$(n)
11220      END FOR n
```

Enfin la dernière ligne DATA contient toutes les terminaisons à chaque personne. Le tableau des terminaisons fonctionne de la même façon que celui des sujets; c'est-à-dire qu'il contient aussi deux dimensions - une pour les entrées et une pour les réponses -, ce tableau est donc dimensionné exactement de manière identique.

```
11135  DATA "E","ES","ES","E","E","E","E","E"
,"ONS","EZ","EZ","ONS","ENT","ENT","ENT","ENT
"
```

```
11230      FOR n=0 TO SU%
11240        READ TE$(n,0) : READ TE$(n,1)
11250      END FOR n
```

CORRESPONDANCE

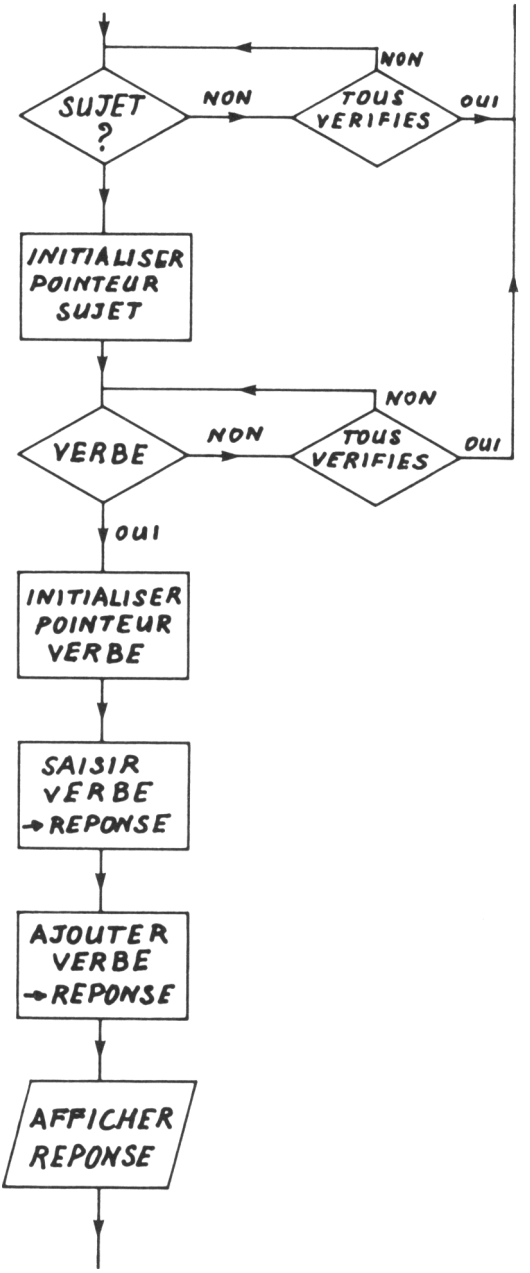
Une fois l'initialisation effectuée, le traitement de la phrase entrée s'opère en plusieurs étapes:

Recherche du sujet: Cette recherche est réalisée comme auparavant, dans le tableau S$(N,0). Si aucun sujet n'est repéré, la fenêtre #0 ne s'efface pas immédiatement.

```
30   REPeat CHAINE
40     AT#0,1,1:CHAINE$=SAISIS$(1)
50     CHAINE$=CHAINE$ & " "
60     SUJ_VERIF
90     FICTIF$=INKEY$(500)
100    CLS#0
110  END REPeat CHAINE

1000 DEFine PROCedure SUJ_VERIF
1010   FOR M=0 TO SU%
1020     IS%=S$(M,0) INSTR CHAINE$
1030       IF IS%>0 THEN
1050         SM%=M
1060         RETurn
1070       END IF
1090   END FOR M
1100     NEXT CHAINE
1110 END DEFine SUJ_VERIF
```

Recherche du verbe: De la même façon que pour le sujet; si un verbe est détecté le pointeur VM% est initialisé à la position où a été détectée la correspondance avec le verbe.

```
70     VERB_VERIF


2000 DEFine PROCedure VERB_VERIF
2010   FOR M=0 TO VB%
2020     IS%=V$(M) INSTR CHAINE$
2030       IF IS%>0 THEN
2040         VM%=M
2060         RETurn
2070       END IF
2080   END FOR M
2090     NEXT CHAINE
2100 END DEFine VERB_VERIF
```



Organigramme 4.3: Initialisation des pointeurs.

REPONSES

Enfin, une fois que le sujet et le verbe ont été identifiés, nous pouvons fournir la réponse appropriée grâce au pointeur VM%.

```
80      REPONSE

3000 DEFine PROCedure REPONSE
3010   RL$=V$(VM%)
```

Le plus simple est de fournir la réponse avec le même sujet que celui qui a été introduit dans la phrase de départ.

```
3060   RL$=S$(SM%,0) & RL$ & TE$(SM%,0) & RE$
(VM%)
3150   PRINT RL$
3190 END DEFine REPONSE
```

Si vous tapez par exemple:

JE CHANTE BIEN

L'ordinateur répondra:

JE CHANTE PARFAITEMENT JUSTE

ou encore:

JE DETESTE LES ORDINATEURS

donne:

JE DETESTE CA PROBABLEMENT AUTANT

D'AUTRES SUJETS

Si vous préférez que la machine vous approuve plutôt que d'essayer de vous battre à votre propre jeu, il suffit de sélectionner la deuxième dimension du tableau S$(N,1), celle qui contient les sujets opposés.

```
3060   RL$=S$(SM%,1) & RL$ & TE$(SM%,1) & RE$
(VM%)
```

Maintenant:

JE CHANTE BIEN

produit la réponse:

TU CHANTES PARFAITEMENT JUSTE

Ce qui est indubitablement plus agréable à lire! Pour plus de suspense, vous pouvez faire tirer un sujet à la machine "au hasard".

```
3005    SUJ=RND(1)
3060    RL$=S$(SM%,SUJ) & RL$ & TE$(SM%,SUJ) &
 RE$(VM%)
```

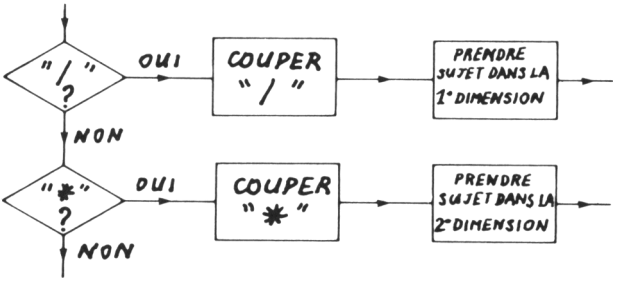
CHOISIR UN SUJET SELON LE CONTEXTE

Il est en fait bien plus logique que le sujet soit choisi en fonction du sens de la phrase. Pour ce faire nous devons inclure des "markers" dans les réponses, qui indiqueront quelle dimension du tableau est à utiliser. Nous utiliserons le signe "/" pour indiquer à la machine que c'est le sujet de la seconde dimension qui doit être utilisé et le signe "*" pour indiquer que c'est dans l'autre dimension qu'il faut aller puiser le sujet de la réponse.

```
11110  DATA "/ CA PROBABLEMENT AUTANT","* DES
 PROGRES FULGURANTS"
11120  DATA "/ D'UN TAS DE CHOSES","* D'UN JO
UR MEILLEUR"
11130  DATA "/ , MAIS RIEN NE DIT QUE CA SE R
EALISERA","* PARFAITEMENT JUSTE","/ BEAUCOUP
MIEUX QU'ON NE POURRAIT LE CROIRE"
```

Bien entendu ni l'un ni l'autre de ces signes ne doit apparaître dans la réponse; il faut donc, lors de l'opération de collage du groupe sujet à la réponse, prendre cette dernière à partir du deuxième caractère (RE$(I$%+1 TO)).

Si aucun "/" n'est détecté, la recherche se continue avec les "*", et le sujet est choisi en fonction de ces éléments.



Organigramme 4.4: Ajuster le sujet au contexte.

```
3000 DEFine PROCedure REPONSE
3010     RL$=V$(VM%)
3030     PRINT#2,RE$(VM%)
3040     IS%="/" INSTR RE$(VM%)
3050       IF IS%>0 THEN
3055         SUJ=1
3060         RL$=S$(SM%,SUJ) & RL$ & TE$(SM%,S
UJ) & RE$(VM%)(IS%+1 TO)
3080       END IF
3090    IS%="*" INSTR RE$(VM%)
3100       IF IS%>0 THEN
3105         SUJ=0
3110         RL$=S$(SM%,SUJ) & RL$ & TE$(SM%,S
UJ) & RE$(VM%)(IS%+1 TO)
3130     END IF
3150   PRINT RL$
3190 END DEFine REPONSE
```

Maintenant:

JE CHANTE BIEN

donne:

JE CHANTE PARFAITEMENT JUSTE

mais:

JE JOUE MAL

renvoie:

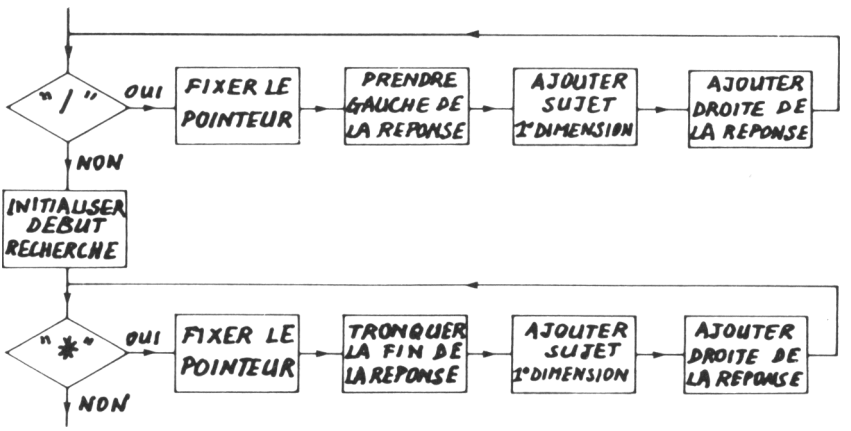
TU JOUES BEAUCOUP MIEUX QU'ON NE POURRAIT LE CROIRE

INSERER LE SUJET DANS LA PHRASE

Pour rendre les choses plus simples nous avons jusqu'ici fait commencer les phrases par le sujet, mais dans la conversation courante ça n'est pas toujours le cas. Etant donné que nous disposons de "markers" pour indiquer quel sujet doit être employé, nous pouvons aussi nous en servir pour insérer les sujets dans la phrase. En premier lieu, il s'agit de modifier les lignes DATA pour pouvoir indiquer à quel endroit les sujets doivent être insérés.

```
11110  DATA "ET ALORS, / CA PROBABLEMENT AUTA
NT","IL EST UN FAIT QUE * DES PROGRES FULGURA
NTS"
11120  DATA "RIEN D'ETONNANT, / D'UN TAS DE C
HOSES","EN EFFET * D'UN JOUR MEILLEUR"
11130  DATA "TOUT CE QUE / NE SE REALISE PAS
NECESSAIREMENT","ON SE DEMANDE SI * PARFAITEM
ENT JUSTE","REFLEXION FAITE / BEAUCOUP MIEUX
QU'ON NE POURRAIT LE CROIRE"
```

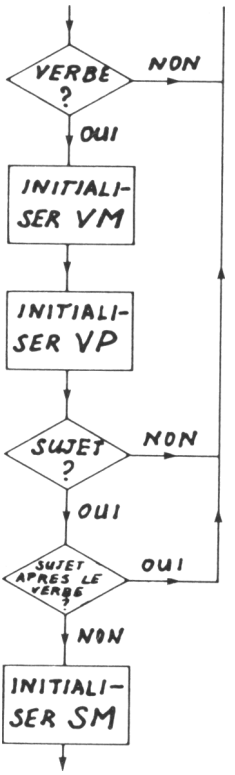
IS% fournit la position où "*" ou "/" ont été trouvés. Tout ce qu'il faut faire est de prendre la partie de la réponse avant ces deux signes (RE$(VM%)(1 TO IS%-1)) puis insérer le sujet S$(SM,SUJ), la racine du verbe RL$, et enfin la terminaison TE$(SM%,SUJ) et le reste de la réponse RE$(VM%)(IS%+1 TO).



Organigramme 4.5: Insertion dans une phrase.

```
3060          RL$=RE$(VM%)(1 TO IS%-1) & S$(SM%
,SUJ) & RL$ & TE$(SM%,SUJ) & RE$(VM%)(IS%+1 T
O)

3110          RL$=RE$(VM%)(1 TO IS%-1) & S$(SM%
,SUJ) & RL$ & TE$(SM%,SUJ) & RE$(VM%)(IS%+1 T
O)
```

Maintenant:
JE CHANTE BIEN
donne:
ON SE DEMANDE VRAIMENT SI TU CHANTES
PARFAITEMENT JUSTE
et:
JE DETESTE LES ORDINATEURS
produit:
ET ALORS, TU DETESTES BEAUCOUP DE CHOSES



Organigramme 4.6: Initialisation des Pointeurs.

CHANGEMENT DE TEMPS

Un des avantages de la méthode des racines, est qu'elle permet de modifier très simplement le programme pour qu'il réponde au passé, au futur etc. En effet, il suffit de changer les terminaisons et le tour est joué.

```
11135  DATA "AIS","AIS","AIS","AIS","AIT","AI
T","AIT","AIT","IONS","IEZ","IEZ","IONS","AIE
NT","AIENT","AIENT","AIENT"
```

5

Les systèmes experts

Un expert est un homme qui a beaucoup de connaissances dans un domaine particulier, et qui peut donner un avis judicieux sur ce sujet.

Une telle capacité ne s'acquiert qu'après une longue formation et une grande expérience, aussi les vrais experts sont-ils rares. De plus ils ne sont pas souvent disponibles lorsqu'un problème doit être résolu.

Des scientifiques se sont posés le problème de créer des programmes qui simulent le comportement de ces experts humains. Ces programmes ont l'avantage de pouvoir être dupliqués très facilement et de produire un grand nombre d'experts qui n'ont besoin ni de pause-café, ni de sommeil, ni d'augmentation de salaire... Dans ce cas l'ordinateur doit être rigoureusement logique, et ne peut bien sûr que suivre les instructions écrites par le programmeur.

Il est intéressant de noter que les auteurs de science-fiction ont envisagé le cas où ces derniers experts se trouvent devant un choix en contradiction avec au moins une de leurs directives, ce qui n'aboutirait pas à un "plantage" du système, mais le conduirait à une "pseudo-dépression nerveuse". (Hal, "A 2001 : Space Odyssey", ou Isaac Asimov, "Positronic robots").

Avant de commencer à écrire des programmes pour systèmes experts, nous devons nous demander

comment travaille un expert humain.

Considérons le cas le plus simple où l'expert doit répondre à un problème connu.

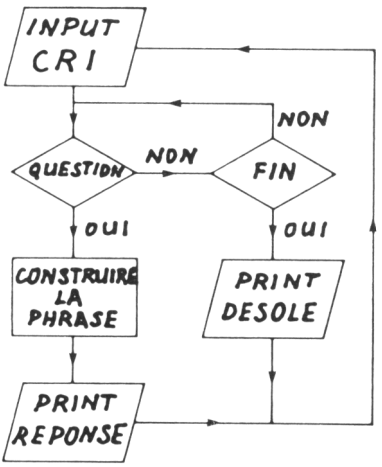
- Il commence par prendre connaissance des données du problème.

- Ensuite il les compare avec celles qu'il a emmagasinées dans son cerveau, et cherche une égalité.

- Enfin il rend compte du résultat.

Dans ce cas nous avons seulement besoin d'un petit programme avec des données, qui essaie de trouver l'égalité entre l'information saisie et celle qui est mémorisée.

Un système convivial utilisera le langage naturel (voir plus loin), mais pour ne pas compliquer les choses nous utiliserons ici une saisie formatée. Pour commencer examinons le problème de la reconnaissance des animaux par leur cri. (Organigramme 5.1).



Organigramme 5.1: Les cris d'animaux.

Nous utiliserons la PROCédure START pour définir deux tableaux: QU$(n), tableau des questions qui contient les cris d'animaux connus, et

RE$(n), tableau des noms d'animaux correspondants.

```
   10 ECRAN
   20 START

10000 DEFine PROCedure START
10010   MODE 4
10020   CLS #0 : CLS #1 : CLS #2
10030 END DEFine ECRAN

11000 DEFine PROCedure START
11010   RESTORE
11020   DIM QU$(4,8), RE$(4,8), IN$(8)
11030   DATA "MIAOU","CHAT","WOUAH","CHIEN","
MEUH","VACHE","HOU","CHOUETTE","HI-HAN","ANE"
11040    FOR N=0 TO 4
11050      READ QU$(N) : READ RE$(N)
11060    END FOR N
11070 END DEFine START
```

Maintenant il suffit de demander un cri d'animal par la fonction SAISIS $(1), et de le comparer avec le contenu de QU$(N). Si l'égalité est trouvée, la PROCédure REPONSE est appelée:

```
30 REPeat QUESTION
40   PRINT \"Quel est l'animal qui fait : ";
50    IN$=SAISIS$(1)
60      FOR N=0 TO 4
70       IF IN$=QU$(N) THEN REPONSE
80      END FOR N
90   PRINT "désolé, je ne sais pas"
100 END REPeat QUESTION

12000 DEFine PROCedure REPONSE
12010   PRINT "L'animal qui fait ";QU$(N);" est
        un(e)",RE$(N)
12020 END QUESTION
12040 END DEFine REPONSE
```

Au point où nous en sommes nous pourrions dire que notre ordinateur remplit mieux cette tâche qu'un expert, car il ne peut porter de jugements subjectifs, ni se lasser ou bien oublier de comparer une donnée de sa mémoire. Par ail-

leurs sa façon d'écrire n'est pas des meilleures lorsqu'il écrit "un(e)". (On laissera le lecteur y remédier au moyen d'une routine qui testera le genre.)

DES BRANCHEMENTS POUR SORTIR

L'exemple ci-dessus est très simple, car il n'y a qu'une question, et une seule réponse possibles. Dans la réalité nous avons à faire à des problèmes beaucoup plus complexes, où la réponse ne peut être obtenue qu'après toute une série de questions.

Que ferait un expert par exemple dans le cas où, ayant mis le contact, sa voiture ne veut pas démarrer?.

Il peut y avoir différentes raisons:

BATTERIE A PLAT
FAUX CONTACTS
CONTACTEUR DEFECTUEUX
DEMARREUR BLOQUE
DEMARREUR MORT
BOBINE HORS D'USAGE

Pour trouver la cause il suivra une démarche logique, et fera un certain nombre de contrôles. Tout d'abord vérifier que seul le démarreur est en panne.

LE VOYANT DU CONTACT EST-IL ALLUME (O/N) ?

Si la réponse est "N", il n'y a pas de courant, et la cause doit être une des trois premières de la liste ci-dessus. Nous pouvons avancer en regardant si les lumières fonctionnent.

LES LUMIERES FONCTIONNENT-ELLES (O/N) ?

Si la réponse est "O", la batterie n'est pas à plat, et il n'y a pas de faux contact, donc l'antivol est probablement en panne, et on peut

proposer de le remplacer.

REMPLACEMENT DU CONTACTEUR

Si les lumières ne fonctionnent pas, il faudra vérifier s'il n'y a pas un faux contact.

LES CONTACTS DE LA BATTERIE SONT-ILS BONS (O/N)?

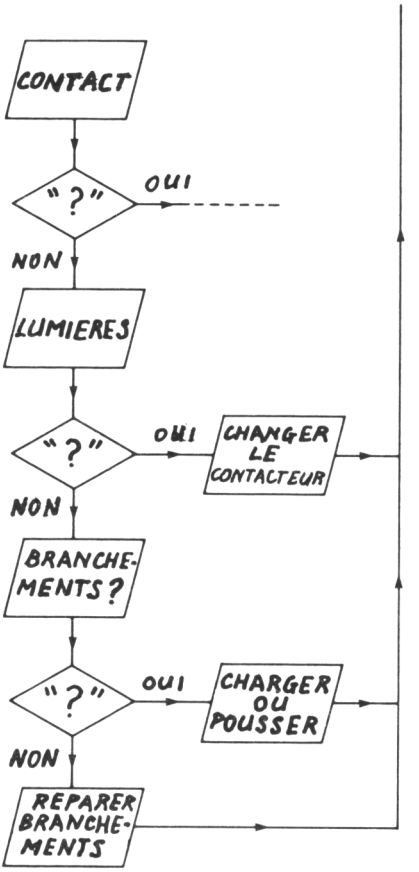
Si la réponse est "O", il faudra la recharger, ou pousser la voiture.

CHARGER LA BATTERIE OU POUSSER

De la même façon, on pourra faire une série de contrôles dans le cas où il y a du courant et que le démarreur ne veut pas fonctionner (les trois dernières possibilités).

La façon la plus simple pour programmer cette suite d'alternatives est d'utiliser une suite de tests "IF ... THEN", qui appellent les PROCedures appropriées à chaque réponse. (Organigramme 5.2).

```
10 ECRAN
20 START
10000 DEFine PROCedure ECRAN
10020   MODE 4
10030   CLS #1 :CLS #2 :CLS #3
10040 END DEFine ECRAN
11000 DEFine PROCedure START
11010   PRINT\"DIAGNOSTIC DE PANNE"
11020   CONTACT
11030 END DEFine START
12000 DEFine PROCedure CONTACT
12010   PRINT\ "LE VOYANT DU CONTACT EST-IL A
LLUME (O/N)";
12020   IN$=SAISIS$(1)
12030    IF IN$="O" THEN FIN
12040   LUMIERES
12050 END DEFine CONTACT
13000 DEFine PROCedure LUMIERES
13010   PRINT\"LES LUMIERES FONCTIONNENT-ELLE
S (O/N)?";
13020   IN$=SAISIS$(1)
13030    IF IN$="O" THEN BATTERIE
```



Organigramme 5.2: Dépannage.

```
13040   PRINT "REMPLACER LE CONTACTEUR"
13050   START
13060 END DEFine LUMIERES
14000 DEFine PROCedure BATTERIE
14010   PRINT\"LES CONNECTIONS AVEC LA BATTER
IE SONT-ELLES OK (O/N) ?";
14020   IN$=SAISIS$(1)
14030    IF IN$="O" THEN CHARGE
14040   PRINT\"REPARER LES CONNECTIONS "
14050   START
14060 END DEFine BATTERIE
15000 DEFine PROCedure CHARGE
```

```
15010   PRINT\"CHARGER LA BATTERIE OU POUSSER
 LA VOITURE "
15020   START
15030 END DEFine CHARGE
16000 DEFine PROCedure FIN
16010   STOP
16020 END DEFine FIN
```

Ce genre de programme est relativement facile à écrire, mais au fur et à mesure qu'il s'allonge et se complique, il devient inutilisable.

VERS UNE SOLUTION

Une solution plus efficace consiste à organiser le texte en tableau avec des pointeurs vers la question ou la réponse suivante, selon la réponse faite à la question posée. (Organigramme 5.3.)

Dans chaque cas le format d'entrée des données est le suivant:

(TEXTE), (pointeur "O"), (pointeur "N").

La première question est:

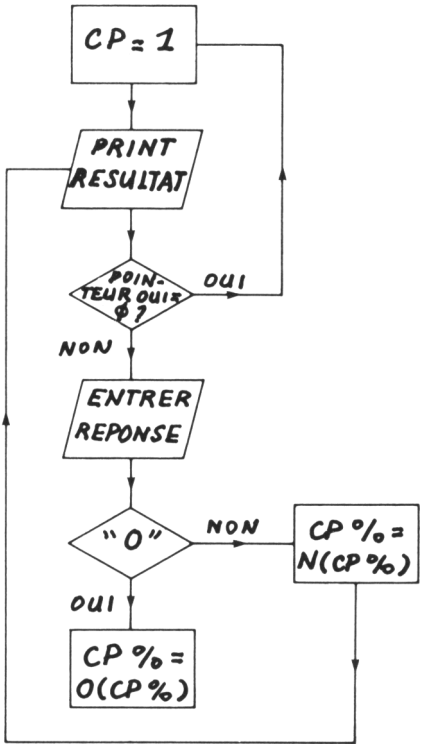
LE VOYANT DU CONTACT EST-IL ALLUME (O/N) ?

Si la réponse est "N", il faut poser la seconde question:

LES LUMIERES FONCTIONNENT-ELLES (O/N) ?

Dans le cas contraire, il faut continuer par l'autre branche de l'organigramme. (Cas 7, non développé.) Nous avons besoin de trois tableaux, TX$(N) qui contiennent le texte en sortie, O(N) pointeur "OUI", N(N) pointeur "NON". Pour pouvoir modifier facilement le programme, nous utiliserons la variable NP%, nombre de groupes de données. Celles-ci sont chargées par groupe de trois dans le poste

correspondant de chacun des tableaux. Les pointeurs vers une sortie du programme sont initialisés à 0. (Remarquer que seule la PROCedure ECRAN est la même que dans le programme précédent, tout le reste du programme est nouveau).



Organigramme 5.3: Pointeur vers la réponse.

```
10 ECRAN
20 START
11000 DEFine PROCedure START
11010   RESTORE
11020   NP%=7:DIM TX$(NP%,30), O(NP%), N(NP%)
11030   DATA "LE CONTACT EST-IL ALLUME (O/N)
?",7,2
11040   DATA "LES LUMIERES MARCHENT-ELLES (O/
N) ?",3,4
11050   DATA "REMPLACER LE CONTACTEUR",0,0
11060   DATA "LES CONNECTIONS AVEC LA BATTERI
E SONT ELLES BONNES (O/N) ?",5,6
```

```
11070   DATA "CHARGER LA BATTERIE OU POUSSER
LA VOITURE (O/N) ?",0,0
11080   DATA "REPARER LES BRANCHEMENTS",0,0
11090   DATA "fin du programme",0,0
11110       READ TX$(M):READ O(M):READ N(M)
11120     END FOR M
11130 END DEFine START
```

Cette routine est très simple. Un pointeur, CP% est utilisé pour indiquer la position courante dans le tableau; il est initialisé à 1 et pointe vers le premier texte. Si cela correspond à une sortie O(CP%)=0, on sort de la boucle QUESTION, et CP% sera réinitialisé à 1 lorsque la PROCedure RESTART sera rappelée. Dans le cas contraire, O(CP%)#0, la boucle QUESTION demande une réponse. Si celle-ci est "O", CP% prend la valeur correspondante du tableau N(N).

```
30 REPeat RESTART
40   UNDER 1:PRINT\"DIAGNOSTIC DE PANNE":UNDE
R 0
50   CP%=1
60     REPeat QUESTION
70       PRINT\ TX$(CP%);" ";
80         IF O(CP%)=0 THEN EXIT QUESTION
90       IN$=SAISIS$(1)
100        IF IN$="O" THEN CP%=O(CP%):NEXT QU
STION
110      CP%=N(CP%)
120    END REPeat QUESTION
130 END REPeat RESTART
```

L'APPROCHE PARALLELE

Après l'approche séquentielle par débranchement utilisée ci-dessus, l'approche parallèle consiste à poser toutes les questions possibles avant de donner un diagnostic. Cette méthode est généralement plus longue que celle qui consiste à suivre une bonne structure en arbre, mais il est plus probable qu'elle donnera la réponse correcte, car aucune comparaison

possible ne sera omise.

Voyons comment nous pourrions distinguer plusieurs moyens de transport.

Nous allons examiner 8 caractéristiques, et indiquer par 1 ou 0 la présence ou non de celles-ci pour cinq modes de transport. (Tableau 5.1). En examinant attentivement on remarque que les résultats sont différents dans chaque cas, il est donc possible de distinguer un moyen de transport par ces critères.

	vélo	auto	train	avion	cheval
roues	1	1	1	1	0
ailes	0	0	0	1	0
moteur	0	1	1	1	0
pneus	1	1	0	1	0
rails	0	0	1	0	0
fenêtre	0	1	1	1	0
chaîne	1	0	0	0	0
direction	1	1	0	1	1

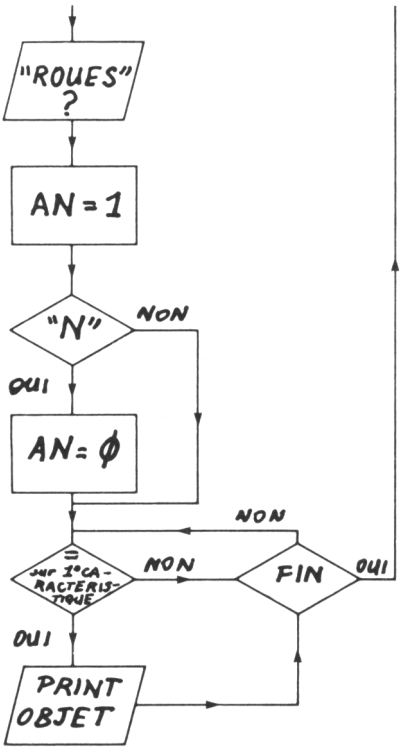
Tableau 5.1: Présence ou non des caractéristiques.

Nous allons entrer ces valeurs en DATA, puis les placer dans un tableau à deux dimensions CA(N,N), qui représentera la table ci-dessus, et un tableau MT$(N), contenant les chaînes de caractères de chacun des mobiles. (Remarquer que ECRAN est inchangé.)

```
   10 ECRAN
   20 START

11000 DEFine PROCedure START
11010   RESTORE
11020   DIM MT$(5), CA(5,8)
11030   DATA "VELO",1,0,0,1,0,0,1,1
11040   DATA "AUTO",1,0,1,1,0,1,0,1
11050   DATA "TRAIN",1,0,1,0,1,1,0,0
10060   DATA "AVION",1,1,1,1,0,1,0,1
10070   DATA "CHEVAL",0,0,0,0,0,0,0,1
10080     FOR N=1 TO 5
10090       READ MT$(N)
```

```
10100          FOR M=1 TO 8
10110          READ CA$(N,M)
10120          NEXT M
10130      NEXT N
11040 END DEFine START
```

Nous pouvons maintenant utiliser les PROCedures QUESTION pour savoir si la première caractéristique est présente ou non, puis COMPARER pour voir quels modes de transport répondent à la question. (Organigramme 5.4).



Organigramme 5.4: Approche parallèle.

```
30 REPeat QUESTION
50   PRINT\"A-T-IL DES ROUES ?";:VERIF
250 END REPeat QUESTION
12000 DEFine PROCedure VERIF
12010   IN$=SAISIS$(1)
12020   AN%=1
```

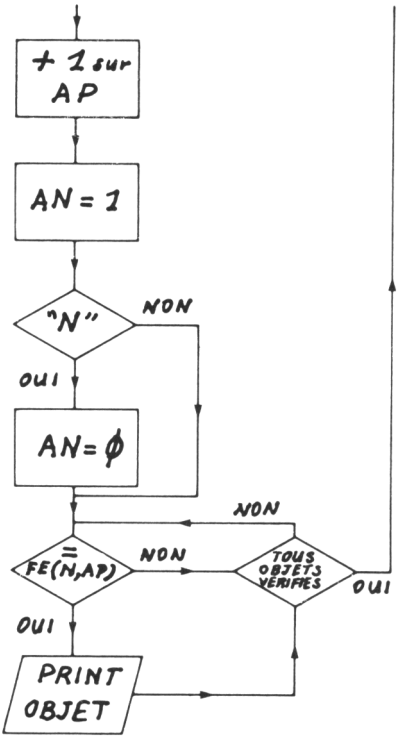
```
12030     IF IN$="N" THEN AN%=0
12040       FOR N=1 TO 5
12050         IF FE(N,1)=AN% THEN PRINT MT$(N)
12060       END FOR N
12070 END DEFine VERIF
```

Dans ce cas une réponse "O" produira l'impression: "VELO, AUTO, TRAIN, AVION". La réponse "N" donnera seulement: "CHEVAL".

Ceci montre clairement un inconvénient possible de la méthode parallèle; en effet, bien que nous venions de démontrer que seul le cheval n'a pas de roue, le programme continue à poser toutes les questions avant de donner sa réponse. Cela n'est pas aussi sot qu'il y paraît au premier abord. En effet on peut répondre "O" à la question suivante (a-t-il des ailes?); on remarquera alors que l'ordinateur, en toute logique, refuse de croire aux chevaux volants.

Utilisons maintenant la PROCedure COMPARER pour examiner les huit caractéristiques dans une même boucle. Il suffit de faire de légères modifications,d'ajouter un pointeur AP%, qui est incrémenté pour comparer l'élément suivant du tableau des caractéristiques CA$(N,AP%). (Organigramme 5.5).

```
30 REPeat QUESTION
50   PRINT\"A-T-IL DES ROUES ";:VERIF
60   PRINT\"A-T-IL DES AILES ";:VERIF
70   PRINT\"A-T-IL UN MOTEUR ";:VERIF
80   PRINT\"A-T-IL DES PNEUS ";:VERIF
90   PRINT\"A-T-IL BESOIN DE RAILS ";:VERIF
130  AP%=0
250 END REPeat QUESTION
11020 DIM MT$(5,7), CA$(5,8):AP%=0
12030 AP%=AP%+1:AN%=1
```



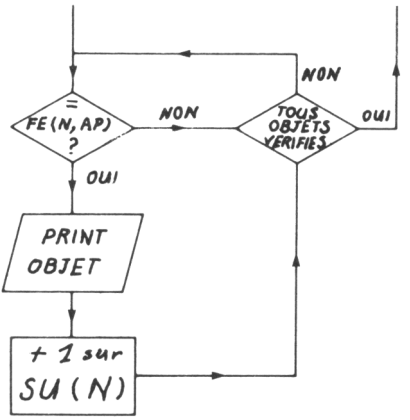
Organigramme 5.5: Comparaisons successives sur les caractéristiques.

DE MIEUX EN MIEUX

La routine ci-dessus donnera successivement après chaque question, la liste des moyens de transport sélectionnés, mais elle ne dira pas quels sont ceux qui répondent à l'ensemble des questions. Modifions-la pour obtenir le nombre de sélection de chacun des moyens de transport. Nous utiliserons un tableau SU(N), dont chaque poste correspond à un moyen de transport, et sera incrémenté lorsque l'on trouvera l'égalité CA(N)=AN%. (Organigramme 5.6).

```
200 PRINT \"RESULTAT"
210  FOR N=1 TO 5
```

```
220     PRINT CA$(N),SU(N)
230     SU(N)=0
240  END FOR N
```



Organigramme 5.6: Recherche de SU(N).

Si un moyen de transport répond à tous les critères,SU(N) sera égal à 8. Si un ou plusieurs critères ne sont pas remplis, il sera diminué d'autant. Il est particulièrement intéressant de mesurer le résultat SU(N) lorsqu'il s'agit de critères subjectifs. (Peut-on parler de la direction d'un cheval?) On peut penser que de toutes façons le résultat le plus fort indiquera la bonne réponse. (Noter que dans ce cas chacune des bonnes réponses a le même poids.)

ENCORE MIEUX AVEC DES CHAINES DE BITS

Vous avez sans doute remarqué que nous avons utilisé 8 critères, il vous est peut-être venu à l'idée que cela n'était pas entièrement fortuit, car il y a huit bits dans un octet. Si nous représentons chaque caractéristique par un nombre binaire, chacun des moyens de transport peut être décrit par un seul nombre décimal somme de ces nombres binaires, au lieu d'une suite de 8 valeurs séparées. Nous ferons la

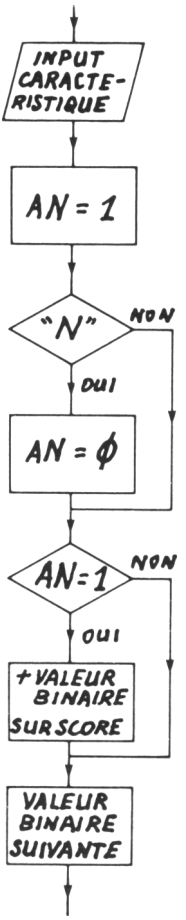
transformation en décimal en prenant comme bit de poids faible celui de la ligne du haut (les roues). Chacune des caractéristiques vaut 1, 2, 4, 8, 16, 32, 64, 128, en notation décimale. (voir Tableau 5.2).

	vélo	auto	train	avion	cheval
roues	1	1	1	1	0
ailes	0	0	0	2	0
moteur	0	4	4	4	0
pneus	8	8	0	8	0
rails	0	0	16	0	0
fenêtre	0	32	32	32	0
chaîne	64	0	0	0	0
direction	128	128	0	128	128
Total	201	173	53	175	128

Tableau 5.2: Des poids binaires.

Il ne sera pas très difficile de remplacer notre résultat de 1 à 8, par le nombre binaire qui convient, en nous souvenant que la valeur décimale du nombre binaire double en passant à la caractéristique suivante, et que nous devons l'ajouter au résultat si AN%=1. (Organigramme 5.7).

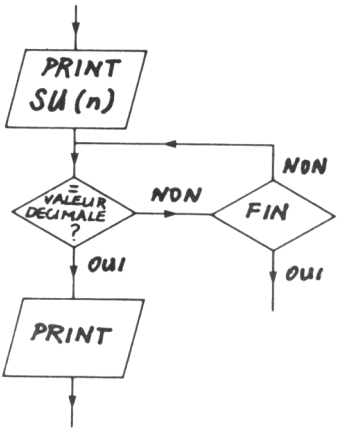
En regardant de plus près, on s'aperçoit qu'il suffit de conserver le nombre total SU%, obtenu en additionnant les valeurs binaires des bonnes réponses. Plus besoin de boucle, ni de comparaison avec chaque poste du tableau, ni même de tableau à deux dimensions, les seules données nécessaires sont les chiffres décimaux, DV(N), qui décrivent chaque moyen de transport. Puis après avoir posé toutes les questions, on comparera ces chiffres à la valeur décimale, SU%, obtenue par transformation des nombres binaires donnés par les réponses. (Organigramme 5.8).



Organigramme 5.7: Résultat en utilisant les nombres binaires.

```
11020 DIM MT$(5,7),DV(5)
11030 DATA "VELO",201
11040 DATA "AUTO",173
11050 DATA "TRAIN",53
11060 DATA "AVION",175
11070 DATA "CHEVAL",128
11090   READ CA$(N)
11100   READ DV(N)
11110   REMark ANNULEE
11120   REMark ANNULLEE
```

```
11140   IF AN%=1 THEN SU%=SU%+BV%
11150   BV%=BV%+BV%
11160   REMark ANNULEE
```



Organigramme 5.8: Comparaison des chiffres à la valeur décimale.

Il est évident que cette méthode fait gagner beaucoup de temps et de place-mémoire, car le tableau doit être mis en place avant toute comparaison, et chaque élément ne demande que quelques octets. Ceci est particulièrement intéressant lorsqu'il y a beaucoup d'informations à traiter. La méthode n'exige pas le calcul a priori des nombres décimaux équivalents à chaque configuration. En outre elle ne donne aucune indication dans le cas d'objet ne répondant pas à tous les critères. (Remarquez que l'on ne peut pas prendre le nombre décimal le plus proche, car la valeur décimale correspondant à chaque critère dépend de sa position.)

On peut toujours faire les calculs à la main, mais il est plus facile de définir une FONCtion BINaire qui les effectuera. On imprimera comme repère une ligne de 8 points, puis on entrera la chaîne de bits, qui sera transmise à BIN. Cette fonction découpe la chaîne à partir de la droite (poids faibles), et si le bit est à 1, la valeur décimale DV% est modifiée. (Noter que le superBASIC permet la comparaison entre une chaîne et une variable simple.) Lorsque tous

les bits ont été testés, la fonction retourne la valeur décimale DV%.

```
    1 CLS
    2 PRINT "........"
    3 INPUT N$
    4 PRINT BIN(N$)
    5 GO TO 2

30000 DEFine FUNction BIN(N$)
30010   DV%=0:BD%=0
30020     FOR N=8 TO 1 STEP -1
30030       IF N$(N)=1 THEN DV%=DV%+BD%
30040       BD%=BD%+BD%
30050     END FOR N
30060   RETurn DV%
30070 END DEFine BIN
```

6

Un système expert ... qui enrichit les connaissances

Les systèmes experts décrits ci-dessus fonctionnent bien, mais ils exigent que vous leur donniez par avance les règles correctes pour prendre leurs décisions, ce qui est astreignant, et même quelquefois impossible lorsque vous n'êtes pas sûr de la réponse.

Par contre, il est possible de construire un programme expert qui tirera un enseignement de ses erreurs et en déduira des règles de décision, ce que les experts essayent de faire. La seule chose nécessaire est de lui dire quand il se trompe, mais pas en quoi. C'est évidemment un avantage si vous n'êtes pas vous même sûr des règles à appliquer. Dans ce cas nous partons avec une série de caractéristiques (critères) qui doivent nous permettre de distinguer entre eux différents objets (sorties), sans établir pour nous guider un tableau de présence ou non de celles-ci (règles de décision). Le programme construit lui-même ce tableau.

Nous allons utiliser notre exemple sur les moyens de transport et commencer par initialiser les variables. FE% est le nombre de caractéristiques à prendre en compte, 8, FE$(N) est un tableau qui contient leur nom, FV(N) un tableau des valeurs attribuées à chacune, et RU(N) un autre qui contient l'ensemble des valeurs de la table de décision pour chaque caractéristique.

```
   10 ECRAN
   20 START

10000 DEFine PROCedure ECRAN
10010   MODE 4
10020   CLS #0 :CLS #1 : CLS #2
10030 END DEFine ECRAN

11000 DEFine PROCedure START
10010   RESTORE
11020   FE%=8 : DIM FE$(FE%,8),FV(FE%),RU(FE%)
11030     FOR N=1 TO FE%
11040       READ FE$(N)
11050     END FOR N
11060   DATA "ROUES","AILES","MOTEUR","PNEUS",
        "RAILS","FENETRES","CHAINE","DIRECTION"
11070 END DEFine START
```

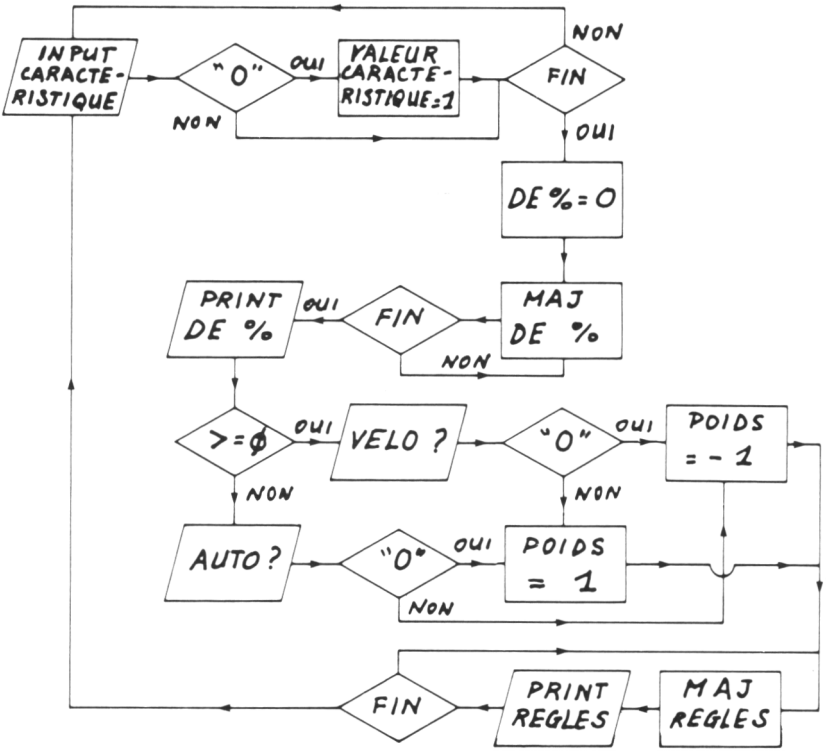
On examine les caractéristiques tour à tour dans la PROCedure QUESTION (Organigramme 6.1). La valeur de la caractéristique courante, FV(N) est mis à 0, une question O/N est posée pour chaque caractéristique. Si la réponse IN$ est oui (O), la valeur FE(N) est mise à 1, dans le cas contraire elle reste à 0. Ceci donnera un tableau de 0 et de 1 FV(N), décrivant la sortie.

```
 50 REPeat QUESTION
 60   FOR N=1 TO FE%
 70     FV(N)=0
 80     PRINT !FE$(N);
 90     IN$=INKEY$(-1)
100     IN$=CHR$((CODE(IN$) || 32)-32)
110       IF IN$="O" THEN FV(N)=1
120   END FOR N

200 END REPeat QUESTION
```

(Remarquez que l'on utilise une méthode plus simple que dans la procédure SAISIS $. en testant un seul caractère).

Maintenant dans la PROCedure MAJ_DE la variable de décision DE% est mise à 0, puis on y met la somme de la valeur courante de DE% plus la somme des valeurs de chaque critère FE(N), multipliée par les valeurs de la table de déci-

sion RU(N).



Organigramme 6.1: Apprendre à distinguer deux objets.

DUQUEL S'AGIT-IL?

Au départ nous allons prendre un cas simple avec deux possibilités (vélo ou auto). Décidons arbitrairement que si la valeur finale de DE%

est égale ou supérieure à 0, ce sera un vélo, et que si celle-ci est inférieure à 0 ce sera une auto. Peu importe si cela est faux, le programme le rectifiera. Quand celui-ci a pris une décision sur la base de la valeur de DE%, il y a confirmation ou infirmation du résultat.

```
180     IF DE%>=0 THEN PRINT\"EST-CE UN VELO "
;:IN$=INKEY$(-1):IN$=CHR$((CODE(IN$) || 32)-3
2):PRINT IN$: VELO
190     IF DE%<0 THEN PRINT\"EST-CE UNE AUTO "
;:IN$=INKEY$(-1):IN$=CHR$((CODE(IN$) || 32)-3
2): PRINT IN$: VOIT
```

Selon que l'on confirme ou non le choix de l'ordinateur, il y a trois possibilités.

S'il est correct, il n'y a rien à faire, la variable de pondération WT% est mise à 0.

Si DE% est >= 0, et si l'ordinateur s'est trompé (il a choisi l'auto), il faut mettre WT% à -1.

Si DE% est < 0, et si l'ordinateur s'est trompé (il a choisi le vélo), il faut mettre WT% à +1.

```
13000 DEFine PROCedure VELO
13010   IF IN$="O" THEN WT%=0 : MAJ_REGLE
13020   WT%=-1 : MAJ_REGLE
13030 END DEFine VELO
14000 DEFine PROCedure VOIT
14010   IF IN$="O" THEN WT%=0 : MAJ_REGLE
14020   WT%=1:MAJ_REGLE
14030 END DEFine VOIT
```

La variable de pondération joue son rôle dans la PROCedure MAJ_REGLE qui modifie les valeurs du tableau RU(N), en les minorant si elles sont trop fortes et en les majorant dans le cas contraire.

```
15000 DEFine PROCedure MAJ_REGLE
15010   PRINT \"REGLES"\
15020   FOR N=1 TO FE%
```

```
15030      RU(N)=((RU(N)*3+(FV(N)*WT%)*3)/3
15040      PRINT RU(N),FE$(N)
15050    END FOR N
15060 END DEFine MAJ_REGLE
```

Pour vérifier le fonctionnement, faisons une démonstration; tapez RUN, puis les entrées ci-dessous. (Remarquez la ponctuation choisie pour que l'affichage indique clairement la relation entre l'entrée et la valeur de la règle).

Entrons:

```
ROUES O      AILES N     MOTEUR N      PNEUS O
RAILS N      FENETRES N CHAINE O       DIRECTION O
```

Le programme restitue une valeur de 0 pour DE%, qui est la valeur initiale et qui n'a pas été modifiée.

```
DE%=0
```

Comme DE% est à 0 le système présume qu'il s'agit du VELO, et demande confirmation. On lui répond bien sûr par "Oui".

```
EST CE UN VELO? O
```

Imprimons le contenu de la table des règles RU(N). On voit que les valeurs sont toutes restées à 0, car par hasard on obtient la bonne réponse!

```
REGLES

  0                    ROUES
  0                    AILES
  0                    MOTEUR
  0                    PNEUS
  0                    RAILS
  0                    FENETRES
  0                    CHAINE
  0                    DIRECTION
```

Entrons maintenant la séquence suivante pour décrire une AUTO:

```
ROUES O      AILES N     MOTEUR O      PNEUS O
RAILS N      FENETRES O CHAINE N       DIRECTION O
```

DE% est toujours à 0, on tombe donc sur la mauvaise conclusion, et la mauvaise question est posée (VELO), il faut donc répondre "Non":

```
DE%=0

EST CE UN VELO? N
```

Comme il y a eu une erreur, la règle de décision est modifiée en soustrayant 1 à toutes les valeurs du tableau où la réponse a été "Oui". Le tableau des règles contient maintenant:

```
REGLES

  -1                ROUES
   0                AILES
  -1                MOTEUR
  -1                PNEUS
   0                RAILS
  -1                FENETRES
   0                CHAINE
  -1                DIRECTION
```

Si l'on entre à nouveau les valeurs pour une AUTO, le programme donnera la bonne réponse.

```
ROUES O      AILES N     MOTEUR O      PNEUS O
RAILS N      FENETRES O CHAINE N       DIRECTION O

DE%=-5

EST CE UNE AUTO? O

REGLES
  -1                ROUES
   0                AILES
  -1                MOTEUR
  -1                PNEUS
   0                RAILS
  -1                FENETRES
   0                CHAINE
  -1                DIRECTION
```

Avant de vous réjouir essayez de lui donner à nouveau les valeurs pour un VELO :il va encore se tromper!

```
ROUES O      AILES N     MOTEUR N      PNEUS O
RAILS N      FENETRES N CHAINE O       DIRECTION O

DE%=-3

EST CE UNE AUTO? N

    REGLES

       0                 ROUES
       0                 AILES
      -1                 MOTEUR
       0                 PNEUS
       0                 RAILS
      -1                 FENETRES
       1                 CHAINE
       0                 DIRECTION
```

Les caractéristiques présentes communes à AUTO et VELO sont majorées de 1 automatiquement, et si nous répétons la dernière phrase on obtiendra la conclusion correcte.

```
ROUES O      AILES N     MOTEUR N      PNEUS O
RAILS N      FENETRES N CHAINE O       DIRECTION O

DE%=1

EST CE UN VELO? O

    REGLES

       0                 ROUES
       0                 AILES
      -1                 MOTEUR
       0                 PNEUS
       0                 RAILS
      -1                 FENETRES
       1                 CHAINE
       0                 DIRECTION
```

La situation est maintenant stable et le programme reconnaîtra toujours correctement AUTO

et VELO, chaque fois que l'on donnera les bonnes caractéristiques.

```
ROUES O      AILES N     MOTEUR O      PNEUS O
RAILS N      FENETRES O  CHAINE N      DIRECTION O

DE%=-2

EST CE UNE AUTO? O

    REGLES

       0                 ROUES
       0                 AILES
      -1                 MOTEUR
       0                 PNEUS
       0                 RAILS
      -1                 FENETRES
       1                 CHAINE
       0                 DIRECTION
```

Remarquez que la valeur finale de DE% est de 1 pour un VELO, et -2 pour une AUTO. Si vous regardez les valeurs du tableau des règles, vous verrez que cela correspond à la fois en valeur et en position aux seules caractéristiques qui distinguent ces deux objets (CHAINE pour VELO, MOTEUR et FENETRE pour AUTO).

GENERALISATION.

Vous avez appris quelque chose à votre ordinateur, mais ce n'est pas extraordinaire d'être capable de distinguer deux objets l'un de l'autre. Etendons le système pour traiter un nombre de possibilités plus grand (Organigramme 6.2).

Pour commencer, définissons le nombre d'objets (sorties) que nous voulons reconnaître, OB%, mettons-les en DATA, qui sont lues dans un nouveau tableau OB$(OB%). Remplaçons notre tableau de décision par un autre à deux dimensions RU(FE%,OB%), qui contient les règles pour chaque objet, puis nous initialisons un

tableau de décision DE(N), pour conserver les valeurs de décision de chaque objet.

```
   20 START

11000 DEFine PROCedure START
11010   RESTORE
11020   FE%=8 : OB%=5 : DIM FE$(FE%,8),FV(FE%),
        RU(FE%,OB%),OB$(OB%,7),DE(OB%) : TS%=5
11030     FOR N=1 TO FE%
11040       READ FE$(N)
11050     END FOR N
11060   DATA "ROUES","AILES","MOTEUR","PNEUS",
        "RAILS","FENETRE","CHAINE","DIRECTION"
11070     FOR N=1 TO OB%
11080       READ OB$(N)
11090     END FOR N
11100   DATA "VELO","AUTO","AVION","TRAIN",
        "CHEVAL"
11110 END DEFine START
```

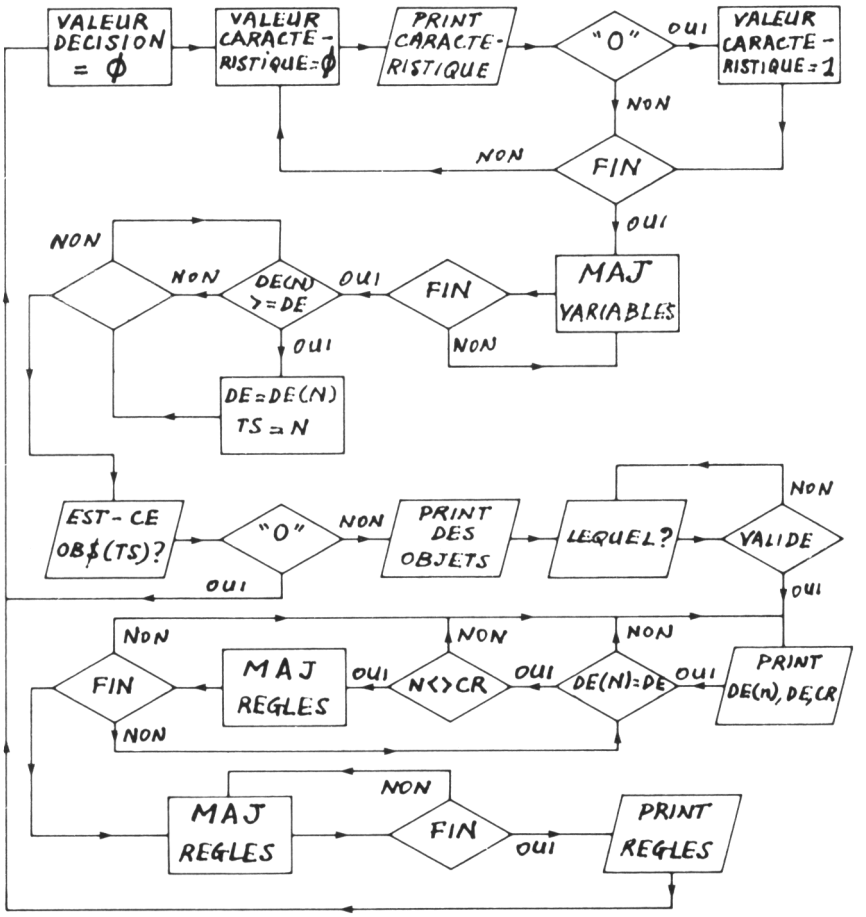
Au lieu d'une variable de décision unique DE%, il nous faut une valeur pour chaque objet. A chaque cycle il faudra mettre à 0, d'abord DE%, puis tous les éléments du tableau décision DE(N), de façon à commencer le traitement de chaque objet dans une situation claire (ZERO_DE).

```
   50    ZERO_DE

14000 DEFine PROCedure ZERO_DE
14010   DE%=0
14020     FOR N=1 TO OB%
14030       DE(N)=0
14040     END FOR N
14050 END DEFine ZERO_DE
```

Les questions sur les valeurs de chaque caractéristique sont ensuites entrées comme ci-dessus.

```
40   REPeat QUESTION

60       FOR N=1 TO FE%
70         FV(N)=0
80         PRINT !FE$(N);
```

```
 90          IN$=INKEY$(-1)
100          IN$=CHR$((CODE(IN$) || 32)-32)
110          PRINT !IN$
120             IF IN$="O" THEN FV(N)=1
130       END FOR N

200    END DEFine QUESTION
```



Organigramme 6.2: Apprendre les règles, généralisation.

MAJ_DE met maintenant à jour chaque élément du tableau de décision, DE(N), selon l'état des valeurs entrées FV(N) et le contenu des éléments correspondants du tableau des règles RU(N,M).

```
  140      MAJ_DV

15000 DEFine PROCedure MAJ_DV
15010   FOR N=1 TO FE%
15020     FOR M=1 TO OB%
15030       DE(M)=((DE(M)*3)+((FV(N)*
            RU(N,M))*3))/3
15040     END FOR M
15050   END FOR N
15060 END DEFine MAJ_DE
```

Il faut maintenant regarder si les valeurs de décision de chaque objet DE(N) sont supérieures ou égales à DE%. Dans ce cas nous conservons N dans un compteur TS%, qui indiquera l'objet pour lequel le nombre de comparaisons est le plus fort.

```
  150      DECISION

16000 DEFine PROCedure DECISION
16010   FOR N=1 TO FE%
16020     IF DE(N)>=DE% THEN DE%=DE(N):TS%=N
16030   END FOR N
16040 END DEFine DECISION
```

Le système fait d'abord la supposition qu'il s'agit de la bonne réponse, il demande confirmation, et si la réponse est affirmative il ne change rien et pose la question suivante.

```
  160      REPONSE

17000 DEFine PROCedure REPONSE
17010   PRINT \"EST CE ";OB$(TS%);" ";
17020   IN$=INKEY$(-1)
17030   IN$=CHR$((CODE(IN$) || 32)-32)
17040   PRINT IN$
17050     IF IN$="O" THEN NEXT QUESTION
17060 END DEFine REPONSE
```

Dans le cas où la réponse est mauvaise le nom et le numéro de chaque objet sont affichés, et l'ordinateur vous demande le numéro de la bonne réponse CR%. (Des bornes sur CR% éviteront d'entrer des valeurs illégales).

```
  170      CORRECTION

18000 DEFine PROCedure CORRECTION
18010   FOR N=1 TO OB%
18020     PRINT \N,OB$(N);
18030   END FOR N
18040   PRINT \\"QUEL NUMERO? ";
18050   IN$=INKEY$(-1)
18060   CR%=CODE(IN$)-48 : IF CR%<1 OR CR%>5
        THEN CORRECTION
18070   PRINT CR%
18080 END DEFine CORRECTION
```

Dans MAJ_REGLES nous devons d'abord voir si la valeur de décision de chaque objet DE(N) est supérieure ou égale à DE%, ET si l'objet considéré N'EST PAS la bonne réponse. Si c'est le cas les règles sont à nouveau mises à jour par soustraction des valeurs des caractéristiques correctes, FE(N).

```
  180      MAJ_REGLES

19000 DEFine PROCedure MAJ_REGLES
19010   FOR N=1 TO OB%
19020     IF DE(N)=DE% AND N<>CR% THEN
19030       FOR M=1 TO FE%
19040         RU(M,N)=((RU(M,N)*3)-
              (FV(M)*3))/3
19050       END FOR M
19060   ELSE NEXT N
19070     END IF
19080   END FOR N
```

Puis les valeurs des caractéristiques correctes sont ajoutées pour effectuer le changement.

```
19090   FOR M=1 TO FE%
19100     RU(M,CR%)=((RU(M,CR%)*3+
          (FV(M)*3))/3
19110   NEXT M
19120 END DEFine MAJ_REGLES
```

Enfin AFF_REGLES affiche l'état du tableau des règles, de façon à ce que l'on y voie plus clair.

```
  190      AFF_REGLES

20000 DEFine PROCedure AFF_REGLES
20010   CLS #2 : CLS #3
20020   FOR M=1 TO OB%
20030     AT #2,3,M-1 : PRINT #2,DE(M);" ";
          DE%;"            ";CR%
20040      FOR N=1 TO FE%
20050        AT #3,(N*3)-3,M-1 : PRINT
             #3,RU(N,M);
20060      END FOR N
20070    END FOR M
20080    PRINT
20090 END DEFine AFF_REGLES
```

Pour rendre l'ensemble du programme plus compréhensible nous allons utiliser les possibilités multi-fenêtre du QL (Figure 6.1). On utilisera les PROCedures AFFICHAGE et LABEL. On passera sur les détails de présentation, l'action principale se passera dans la fenêtre par défaut (moitié droite de l'écran), avec l'affichage de DE(N), DE% et CE% dans la fenêtre #2, les règles dans la fenêtre #3, et des intitulés divers dans les fenêtres #4, #5, et #6.

Remarquez qu'une procédure ECRAN est définie pour effacer l'écran au départ et donner un filet de sécurité qui vous permet de lister le programme. Les deux paramètres passés à ECRAN sont PAPER et INK. La frappe de ECRAN 6,0 en commande directe avant LIST donnera le listing en blanc sur fond noir sur tout l'écran.

```
   10 ECRAN 0,6 :  AFFICHAGE
   30 LABEL

10000  DEFine  PROCedure AFFICHAGE
10010    MODE 0
10020    CLS  #0
10030      INK  #0,7
10100    WINDOW #1,230,200,257,16
10110      BORDER #1,3,6
10120      CSIZE #1,2,0
10130      PAPER #1,5
10140      INK #1,0
```

```
10150      CLS #1
10200    WINDOW #2,140,50,105,32
10220      CSIZE 352,0,0
10240      INK #2,2
10250      CLS #2
10300    OPEN #3,SCR_170X60A85X100
10310      BORDER #3,3,2
10320      CSIZE #3,0,0
10330      PAPER #3,6
10340      INK #3,0
10350      CLS #3
10400    OPEN #4,SCR_175X90A80X155
10410      BORDER #4,5,2
10420      CSIZE #4,0,0
10430      INK #4,1,0
10440      PAPER #4,6
10450      CLS #4
10500    OPEN #5,SCR_230X70A26X90
10510      BORDER #5,3,2
10520      CSIZE #5,0,0
10530      INK #5,4
10540      PAPER  #5,6
10550      CLS #5
10600    OPEN #5,SCR_230X70A26X18
10610      BORDER #6,5,2
10620      CSIZE #6,0,0
10630      INK #6,1,4
10640      PAPER #6,6
10650      CLS #6
10700 END DEFine AFFICHAGE

20000 DEFine PROCedure ECRAN (A,B)
20010    WINDOW, #2,460,200,26,16
20020    PAPER #2,A
20030    INK #2,B
20040    CLS #2
20050 END DEFine ECRAN

13000 DEFine PROCedure LABEL
13010    AT #6,11,0 : CSIZE #6,1,0 : PRINT
         #6,"DE(N)    DE%    CR%" : CSIZE #6,0,0
13020      FOR N=1 TO OB%
13030        AT #6,1,N : PRINT #6,OB$(N)
13040        AT #5,1,N : PRINT #,OB$(N)
13050      END FOR N
13060      FOR N=1 TO FE%
13070        FOR M=1 TO 8
```

```
13080          AT #4,(N*3)-2,M-1
13090          PRINT #4,FE$(N,M)
13100          AT #5,15,0 : CSIZE #5,3,0 : PRINT
               #5,"REGLES" : CSIZE #5,0,0
13110         END FOR M
13120       END FOR N
13130 END DEFine LABEL
```

Une fois encore une démonstration vous fera comprendre ce qui se passe lorsque l'on entre la suite ci-dessous:

```
ROUES O                AILES N
MOTEUR N               PNEUS O
RAILS N N              FENETRE N
CHAINE O               DIRECTION O
```

Le programme arrive à la conclusion erronée qu'il s'agit du CHEVAL, aussi il faut lui dire que c'est faux et lorsqu'il demande la bonne réponse donner VELO (1).

```
EST-CE CHEVAL? N

1            VELO
2            AUTO
3            TRAIN
4            AVION
5            CHEVAL

LEQUEL EST-CE? 1
```

L'état des différents tableaux est affiché pour information dans les fenêtres de gauche.

	DE(N)	DE%	CR%
AUTO	0	0	1
TRAIN	0	0	1
AVION	0	0	1
CHEVAL	0	0	1

REGLES

VELO	1	0	0	1	0	0	1	1
AUTO	-1	0	0	-1	0	0	-1	-1
TRAIN	-1	0	0	-1	0	0	-1	-1

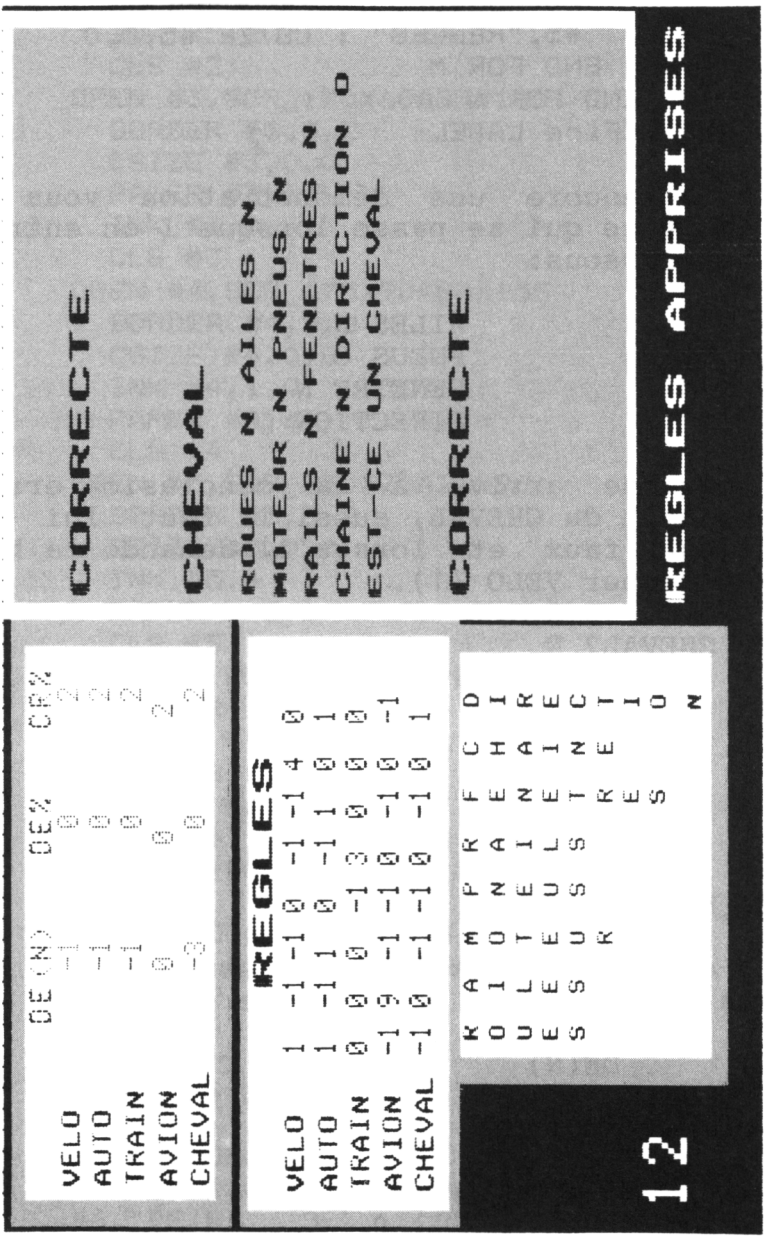


Figure 6.1: Un système expert enrichi.

	ROUES	AILES	MOTEUR	PNEUS	RAILS	FENETRES	CHAINE	DIRECTION
AVION	-1	0	0	-1	0	0	-1	-1
CHEVAL	-1	0	0	-1	0	0	-1	-1

A l'examen, vous verrez que les caractéristiques qui ont produit des changements dans les règles sont les roues, les pneus, la chaîne et la direction. Ce sont toutes les caractéristiques du VELO, qui ne se trouvent pas chez le CHEVAL. Vous remarquerez aussi que les valeurs de ces caractéristiques sont maintenant à 1 dans le tableau des règles du VELO, et à -1 pour les autres objets. Maintenant, donnons-lui les caractéristiques d'une AUTO, il la prendra pour un VELO; effectuons la correction. Remarquez que le tableau des règles pour VELO et AUTO est modifié pour tenir compte des informations nouvelles.

```
ROUES O                    AILES N
MOTEUR O                   PNEUS O
RAILS N                    FENETRES O
CHAINE N                   DIRECTION O

EST-CE VELO? N

1          VELO
2          AUTO
3          TRAIN
4          AVION
5          CHEVAL

LEQUEL EST-CE? 2

                DE(N)       DE%       CR%

VELO              3          3         2
AUTO             -3          3         2
```

TRAIN	-3	3	2
AVION	-3	3	2
CHEVAL	-3	3	21

REGLES

	ROUES	AILES	MOTEUR	PNEUS	RAILS	FENETRES	CHAINE	DIRECTION
VELO	0	0	-1	0	0	-1	1	0
AUTO	0	0	1	0	0	1	-1	0
TRAIN	-1	0	0	-1	0	0	-1	-1
AVION	-1	0	0	-1	0	0	-1	-1
CHEVAL	-1	0	0	-1	0	0	-1	-1

Essayez ensuite AVION, il répondra AUTO, effectuez la correction.

ROUES O
MOTEUR O
RAILS N
CHAINE N
AILES O
PNEUS O
FENETRES O
DIRECTION O

EST-CE AUTO? N

1 VELO
2 AUTO
3 TRAIN
4 AVION
5 CHEVAL

LEQUEL EST-CE? 4

Et maintenant TRAIN, la réponse est toujours fausse.

ROUES O
MOTEUR O
RAILS O
CHAINE N
AILES N
PNEUS N
FENETRES O
DIRECTION N

```
EST-CE AVION? N

1          VELO
2          AUTO
3          TRAIN
4          AVION
5          CHEVAL

LEQUEL EST-CE? 3

Enfin CHEVAL, il répond AVION.

ROUES N                    AILES N
MOTEUR N                   PNEUS N
RAILS N                    FENETRES N
CHAINE N                   DIRECTION O

EST-CE AVION? N

1          VELO
2          AUTO
3          TRAIN
4          AVION
5          CHEVAL

LEQUEL EST-CE? 5
```

Si vous continuez à rentrer des informations il finira par donner la bonne réponse chaque fois. Cela prendra plus ou moins de temps selon l'importance des différences entre les caractéristiques des objets, et selon leur ordre d'introduction dans le système. Attention cela peut prendre beaucoup de temps! Voici une séquence qui donne le bon résultat.

```
AVION  (TRAIN)  AUTO   (AVION)   VELO   (OUI)
AUTO   (OUI)    AVION  (AUTO)    AVION  (OUI)
CHEVAL (OUI)    AVION  (VELO)    AUTO   (AVION)
AVION  (AUTO)   AVION  (AUTO)    AUTO   (AVION)
AUTO   (OUI)    AVION  (AUTO)    AVION  (OUI)
AUTO   (OUI)    AVION  (OUI)     CHEVAL (OUI)
VELO   (OUI)    TRAIN  (AUTO)    TRAIN  (OUI)
VELO   (OUI)    AUTO   (AVION)   AUTO   (OUI)
AVION  (AUTO)   AVION  (OUI)     AUTO   (AVION)
AUTO   (OUI)    AVION  (OUI)     AUTO   (OUI)
VELO   (AUTO)   AUTO   (OUI)     AVION  (OUI)
TRAIN  (OUI)    CHEVAL (OUI)     VELO   (OUI)
```

Comme l'échelle des valeurs va de +6 à -2, vous ne serez pas surpris du temps nécessaire pour en arriver-là.

REGLES

	ROUES	AILES	MOTEUR	PNEUS	RAILS	FENETRES	CHAINE	DIRECTION
VELO	1	0	-1	1	0	-2	3	0
AUTO	-1	4	1	0	-1	1	-2	0
TRAIN	0	-1	1	-2	2	1	-1	-2
AVION	-2	6	0	0	-1	0	-2	-2
CHEVAL	-1	0	0	-1	0	0	-1	01

DIGESTION AUTOMATIQUE DES DONNEES.

Notre expert finit par trouver les règles lui-même, mais il est fastidieux de faire la conversation avec lui pendant qu'il construit son tableau de règles. Dans la réalité il serait bien préférable de lui donner une masse d'informations sur un sujet, et de le laisser seul digérer le tout et en déduire les règles.

Il n'est pas très difficile de modifier notre programme qui traite les informations des DATAS.

Tout d'abord entrons l'information avec le format suivant, nom de l'objet, suivi de "O" ou "N" pour chaque caractéristique, dans le bon ordre.

```
25000 REMark CARACTERISTIQUES
25010   DATA "VELO","O","N","N","O","N","N","
O","O"
```

```
25020   DATA "AUTO","O","N","O","O","N","O","N",
        "O"
25030   DATA "TRAIN","O","N","O","N","O","O","N"
        ,"N"
25040   DATA "AVION","O","O","O","O","N","O","N"
        ,"O"
25050   DATA "CHEVAL","N","N","N","N","N","N",
        "N","O"
25060   DATA "FIN"
```

Introduisons une PROCédure LECTURE, appelée au début de la boucle QUESTION. Pour le moment elle lit et imprime les noms (N$)des objets examinés.

```
45 LECTURE

26000 DEFine PROCedure LECTURE
26030   READ N$
26100   CSIZE 3,1 : PRINT N$ : CSIZE 2,0
26110 END DEFine LECTURE
```

Les "O" et "N" correspondant à chaque caractéristique sont lus dans IN$, à la place du INKEY$ dans la boucle QUESTION.

```
90 READ IN$
```

Dans la PROCedure REPONSE, nous comparons le nom de l'objet examiné (N$), avec celui de l'objet qui a le meilleur résultat et choisi par notre expert (OB$(TS%)). En cas d'égalité "CORRECT" est affiché.

```
14020 IF OB$(TS%)=N$ THEN
14030   CSIZE 3,1:PRINT \"CORRECTE"\\:CSIZE 2
,0
14040   NEXT QUESTION
14050 END IF
```

Dans la PROCedure CORRECTION nous comparons le nom de l'objet examiné (N$) avec celui de tous les objets connus de notre expert. La meilleure façon pour le faire est d'insérer une comparaison dans la boucle qui fixe CR% à N lorsqu'il y a égalité. Le INKEY$ d'origine et les tests de validation doivent être supprimés.

```
17025 IF OB$(N)=N$ THEN CR%=N
17050 REMark ANNULEE
17060 REMark ANNULEE
```

Une fois ceci fait, vous pouvez vous reposer ou prendre une tasse de café pendant que votre expert travaille dur!

CA TOURNE

A présent le programme se terminera lorsque tous les objets auront été examinés une fois, ce qui est insuffisant comme vous savez pour construire les règles correctes. On forcera la répétition en testant le message "FIN", et en faisant un RESTORE. Remarquez que nous devons lire à nouveau N$ après le RESTORE.

```
26040 IF N$="FIN" THEN
26050 RESTORE 27000
26080 READ N$
26090 END IF
```

Pour voir comment marche notre expert et pouvoir le féliciter, il faut garder la trace du nombre de cycles de comparaisons effectués, et savoir quand il a réussi. Deux nouvelles variables sont définies. CY% est le nombre de cycles de comparaisons effectués, et SU% le nombre de cycles lorsque le résultat est acquis. SU% est incrémenté dans la PROCedure REPONSE, et réinitialisé au moment du RESTORE, puis comparé au nombre d'objets correctement identifiés (5). Un affichage du cycle courant dans la fenêtre #0 en bas à gauche permet de suivre l'avancement. Un message "REGLES APPRISES" apparaît lorsque SU% atteint 5.

```
5 CY%=1:SU%=0
14040   SU%=SU%+1: NEXT QUESTION
26010   IF SU%=5 THEN STOP
26020  AT#0,0,0:CSIZE #0,3,1 : PRINT #0,CY%:C
SIZE #0,0,0
```

```
26070     SU%=0
```

En regardant cette version automatisée, vous verrez qu'il faut six cycles de DATA pour reconnaître les cinq moyens de transport. Si on permute VELO et CHEVAL, il n'en faut que quatre. Si AVION et VELO sont permutés, on arrive aussi à quatre. Par contre si on échange VELO et AUTO il en faut douze! Il est intéressant de remarquer que les règles finales changent dans chaque cas. On vous laissera essayer un ordre aléatoire pour les DATAS, et étendre le domaine de connaissance.

LA CONSERVATION DE NOTRE EXPERT.

Notre expert a une bonne expérience, il serait dommage de le perdre en coupant le courant. Comme les règles sont conservées en tableaux, vous pourrez facilement écrire une routine pour les sauvegarder et les recharger pour une nouvelle utilisation.

7

Des comparaisons approchées

Les ordinateurs ne sont pas complètement logiques, et notre mémoire est beaucoup moins fiable, ce qui nous donne des soucis lorsque nous cherchons à retrouver des informations sur un sujet donné. L'anglais par exemple est une langue qui présente de multiples formes en particulier pour l'orthographe des mots. Une façon d'envisager le problème est de s'intéresser aux consonances des mots plutôt qu'à leur orthographe par l'utilisation d'un codage, ce qui a été fait pour la première fois pour le recensement de 1890 aux USA. Cette méthode donne un code voisin pour les mots qui ont une consonance proche.

Les règles de codage sont les suivantes:

1) Toujours conserver la première lettre en tête du code, puis à partir de la seconde:
2) Ignorer les voyelles a,e,i,o,u.
3) Ignorer les lettres w,y,q,h.
4) Ignorer les marques de ponctuation.
5) Coder les autres lettres avec les valeurs 1 à 6:

b,f,p,v	=	1
c,g,j,k,s,x,z	=	2
d,t	=	3
l	=	4
m,n	=	5
r	=	6

6) Lorsque deux lettres adjacentes ont le même code, ne garder que le premier.
7) Si la longeur du code est supérieure à 4, ne garder que quatre caractères.

8) Si la longueur du code est inférieure à 4, compléter à gauche par des zéros.

Donnons quelques exemples:

BRAIN - B650: B est gardé, R vaut 6, A et I sont sautés, N vaut 5, et un 0 est ajouté.

CUNNINGHAM - C552: C est gardé, U est sauté, NN donne 5, I est sauté, N vaut 5, G vaut 2, H et A sont sautés, M vaut 5. Le résultat C5525 est tronqué à 4 caractères.

GORE - G600: G est conservé, O est sauté, R vaut 6, E est sauté, deux 0 sont ajoutés.

IRELAND - I645: I est gardé, R vaut 6, E sauté, L vaut 4, A est sauté, D vaut 5 . Le résultat I6453 est tronqué à 4 caractères.

SCOT - S300: S est gardé, C est sauté car il a la même valeur que S, O est sauté, T vaut 3, et un 0 est ajouté.

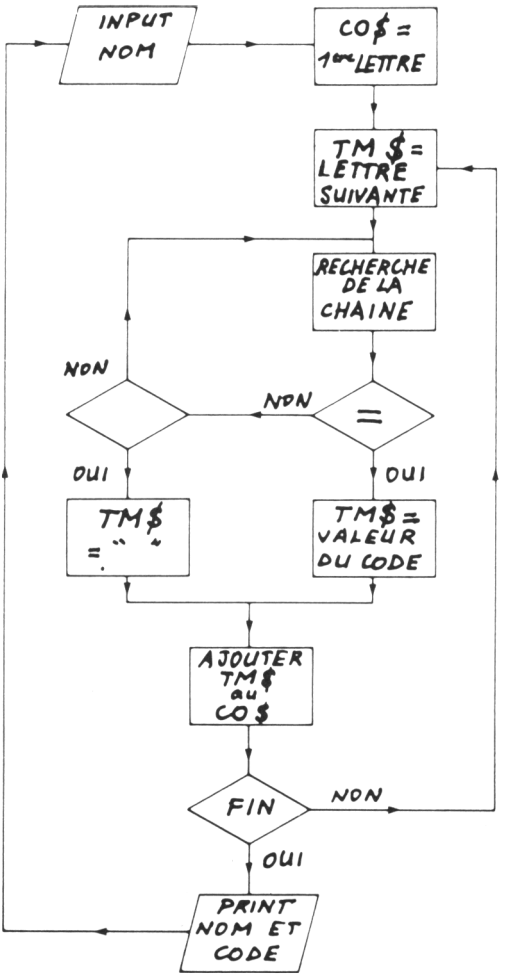
Si votre nom ne contient que des voyelles et des lettres rejetèes, votre code sera court!

HEYHOE - H0000: H est retenu, les autres lettres rejetées, quatre 0 sont ajoutés.

ROUTINE DE CODAGE.

Ecrivons un petit programme qui permet de trouver le code d'un mot (Organigramme 7.1). Commençons par une routine d'initialisation qui restaure les DATA et appelle ECRAN pour la mise en place d'une série de fenêtres.

```
10 INITIALISATION
10000 DEFine PROCedure INITIALISATION
10010     RESTORE
10020     ECRAN
10030     CODES
10050 END DEFine INITIALISATION
```



Organigramme 7.1: Méthode de codage.

L'écran est divisé verticalement en deux fenêtres principales 1, et 2, puis 0 est réservée à la saisie, et 3 et 4 pour des libellés. (voir Figure 7.1).

```
11000 DEFine PROCedure ECRAN
11010   MODE 4
11020   WINDOW #2,230,186,26,30
11030   BORDER #2,2,6
11040   CSIZE 2,1,0
11050   CLS #2
```

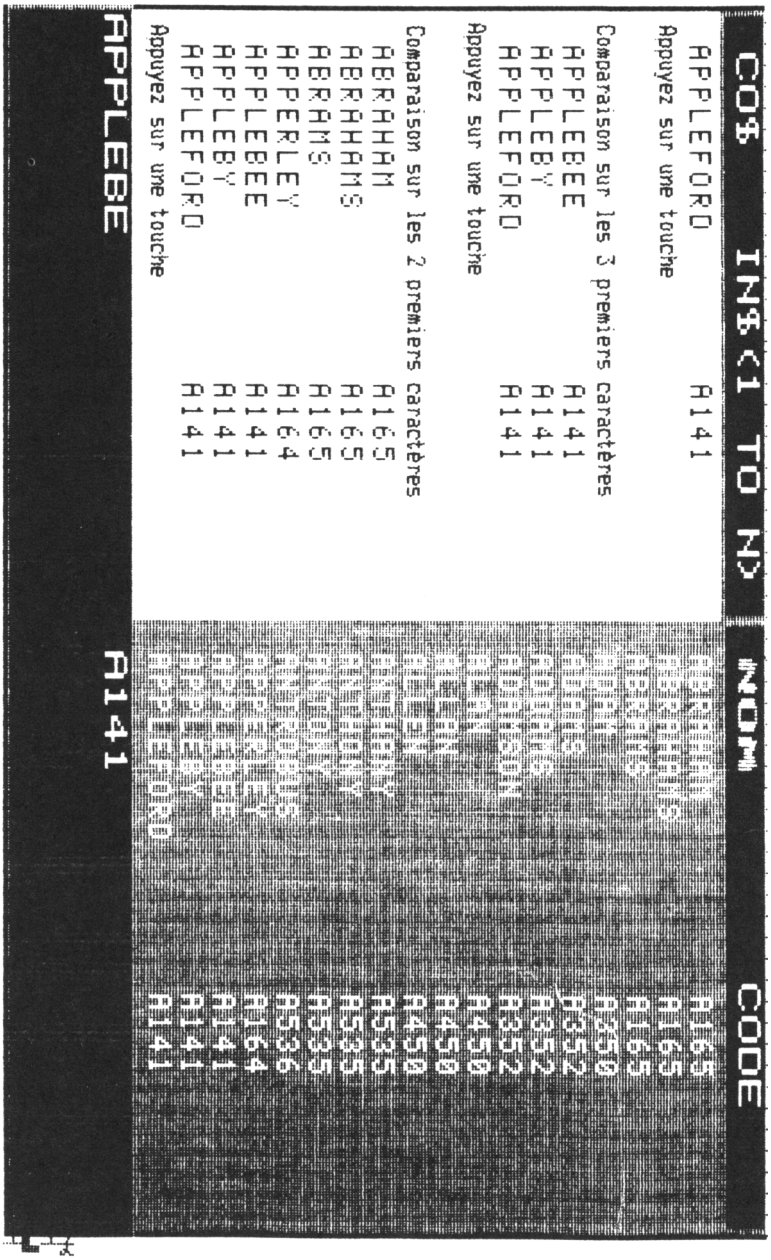


Figure 7.1: Comparaisons approchées.

```
11060   WINDOW #2,230,186,26,30
11070   BORDER #2,2,6
11080   CSIZE 2,1,0
11090   PAPER #1,5
11100   INK #1,0
11110   CLS #1
11120   BORDER #0,1,4
11130   CSIZE #0,2,0
11140   CLS #0
11150   OPEN #3,SCR_230X14A257X16
11160   OPEN #4,SCR_230X14A26X16
11170   BORDER #3,1,4
11180   BORDER #4,1,4
11900   CSIZE #3,2,0
11200   CSIZE #4,2,0
11210   PAPER #3,0
11210   PAPER #4,0
11230   INK #3,7
11240   INK #4,7
11250   CLS #3
11260   CLS #4
11270   PRINT #3,"NOM","   CODE"
11280   PRINT #4,"  CO$";"   ";"IN$(1 TO N)";
11290 END DEFine ECRAN
```

La PROCedure CODES lit chaque groupe de lettres retenues dans un élément de tableau, SC$(N). Remarquez que les groupes sont rangés dans l'ordre de la valeur correspondante du code.

```
12000 DEFine PROCedure CODES
12010   DIM SC$(6,7)
12020   DATA "BFPV","CGJKSXZ","DT","L","MN","R"
12030     FOR N=1 TO 6
12040       READ SC$(N)
12050     END FOR N
12060 END DEFine CODES
```

Nous pouvons maintenant saisir un mot à coder, IN$. (On se protège d'une chaîne vide, et si vous voulez effectuer la conversion à partir des minuscules vous pouvez utiliser la procedure SAISIS$ du chapitre 2.)

```
100 REPeat BOUCLE
110    AT #0,1,2 : INPUT #0,IN$;: IF IN$="" T
```

```
HEN BOUCLE
140 END REPeat BOUCLE
```

Comme nous allons utiliser en différents endroits ce codage, nous allons en faire une FONCTION nommée COMPILE$. Ceci permettra d'afficher simplement le résultat.

```
120 AT #0,20,2 : PRINT #0, COMPILE$(IN$)\\
```

Pour commencer, prenons le premier caractère de IN$ selon la règle 1.

```
15000 DEFine FUNCtion COMPILE$ (IN$)
15010   TM$=IN$(1) : CO$=TM$ : CONVERSION
```

Pour coder chaque lettre il faut comparer TM$ à chacune des lettres de chaque groupe SC$(N). Pour comparer chaque lettre d'un groupe, il faudra, six fois de suite, fixer la chaîne à rechercher SE%, (code courant du groupe), et utiliser une instruction INSTR qui compare chaque lettre du groupe à TM$.

```
16000 DEFine PROCedure CONVERSION
16010   LOCal P
16020     FOR P=1 TO 6
16030       SE$=SC$(P)
16040       SP%=TM$ INSTR SE$
```

Après la comparaison ci-dessus, il faut savoir si il y a eu égalité avec une lettre d'un groupe, et si oui quel groupe. Si on ne trouve pas d'égalité, SP% est mis à 0. Dans le cas contraire SP% est initialisé à P, qui donne la valeur du groupe; on transforme P en chaîne, TM$ qui remplace notre chaîne temporaire précédente.

```
16050     IF SP%>0 THEN TM$=P : RETURN
```

Si aucune égalité n'a été trouvée dans ce groupe, il faut passer au suivant.

```
16060     END FOR P
```

Si aucune égalité n'a été trouvée dans aucun des groupes, TM$ doit contenir un caractère ignoré, et on le remet à vide.

```
16070   TM$=""
16080 END DEFine CONVERSION
```

Il faut ensuite comparer tour à tour les autres lettres du mot, 2 TO LEN(IN$), et utiliser une chaîne temporaire TM$ égale à la lettre à coder.

```
15030   FOR N=2 TO LEN(IN$)
15040     TM$=IN$
15050     CONVERSION
```

On ajoute le nouveau caractère TM$ à la chaîne-code précédente CO$, et on renvoie CO$ lorsque tous les caractères de IN$ ont été examinés.

```
15080         CO$=CO$&TM$

15110     END FOR N

15140     PRINT #2,
15150       RETurn CO$
15160 END DEFine COMPILES
```

Le code final sera affiché en bas de l'écran, à côté de la saisie, mais il serait instructif d'examiner comment on en arrive là. En ajoutant la ligne suivante ,la routine COMPILE$, donnera un affichage détaillé dans la fenêtre de gauche, au fur et à mesure de l'avancement.

```
15100      PRINT #2," ";CO$,,IN$(1 TO N)
```

En entrant STEVEN on obtient:

```
CO$   IN$(1 TO N)
S3    ST
S3    STE
S31   STEV
S31   STEVE
S315  STEVEN
```

De même pour BRAIN et CUNNIGHAM:

```
CO$       IN$(1 TO N)
B6        BR
B6        BRA
B6        BRAI
B65       BRAIN

C         CU
C5        CUN
C55       CUNN
C55       CUNNI
C555      CUNNIN
C5552     CUNNING
C5552     CUNNINGH
C5552     CUNNINGHA
C55525    CUNNINGHAM
```

Le code de BRAIN est trop court et doit être complété avec des 0, tandis que celui de CUNNINGHAM est trop long et on retrouve deux fois le même code après la lettre N.

LA MISE AU POINT DES DETAILS

Pour régler la question de la répétition de code pour deux lettres consécutives il faudra conserver la chaîne précédente temporaire, LT$. Pour initialiser LT$ il nous faut le code du premier caractère de IN$ pour le cas où un mot commence par deux lettres identiques. Dans la boucle il faudra comparer LT$ et TM$, s'il y a égalité il ne faudra pas concaténer TM$ à CO$. Par ailleurs il faut donner à LT$ la dernière valeur de TM$.

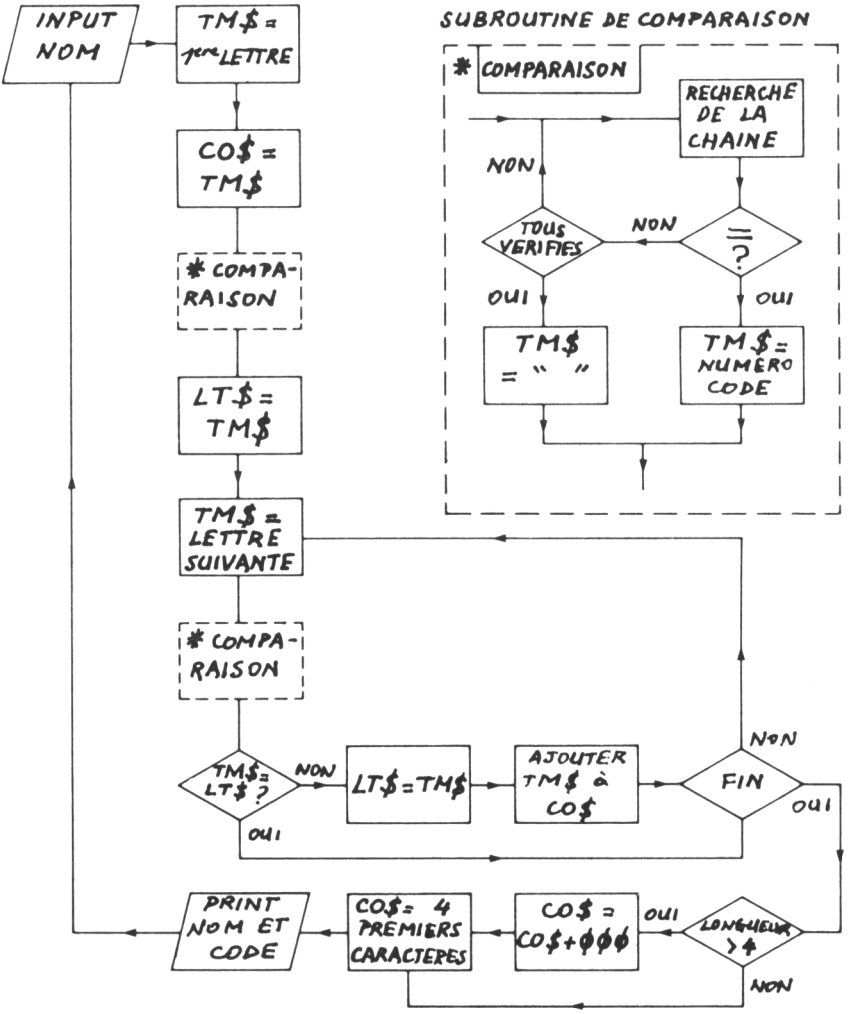
```
15020  LT$=TM$

15060          IF TM$<>LT$ THEN
15070            LT$=TM$
15090          END IF
```

Réglons maintenant le cas du code trop court. Après un test sur la longueur, on concatène 3 zéros et on tronque à la bonne

longueur (4 caractères).

```
15120          IF LEN(CO$)<4 THEN CO$&"000" :
               CO$=CO$(1 TO 4)
```

Enfin si le code est trop long on le tronque à une longueur de 4. (Organigramme 7.2).



Organigramme 7.2: Les détails.

```
15130          IF LEN(CO$)>4 THEN CO$=CO$(1 TO 4)
```

UTILISATION

Nous avons maintenant une méthode fiable pour effectuer le codage et nous allons l'essayer. Lisons d'abord une série de noms placés en DATA, et rangeons-les dans NA$(N). Notre liste d'essai contient 18 noms, mais un coup d'oeil dans l'annuaire du téléphone vous en fournira beaucoup plus si vous le désirez. Remarquez que le nombre de mots est mis dans NW%, et que cette PROCedure est appelée depuis INITIALISATION.

```
10040    NOMS

13000 DEFine PROCedure NOMS
13010   LT$="" : NW%=17
13020   DIM NA$(NW%,16) : DIM NC$(NW%,16)
13030   DATA "ABRAHAM","ABRAHAMS","ABRAMS",
        "ADAMS","ADDAMS","ADAMSON","ALAN"
        "ALLAN","ALLEN"
13040   DATA "ANTHANY","ANTHONY","ANTONY",
        "ANTOBUS","APPERLEY","APPLEBEE",
        "APPLEBY","APPLEFORD"
13050     FOR N=0 TO NW%
13060       READ NA$(N)
13070     END FOR N

13120 END DEFine NOMS
```

L'idée principale consiste à comparer les codes avant d'afficher Les mots possibles. Nous avons besoin des codes de tous les noms en DATA, que nous plaçons dans un tableau de chaînes équivalent NC$(N). Ceci n'est pas difficile car nous utiliserons notre FONCtion COMPILE$ en lui passant NA$(Q) au lieu de IN$.

En lançant l'exécution, on voit la réalisation et l'affichage des codes de chaque donnée respectivement dans les fenêtres de gauche et de droite. Comme la présentation à droite n'est pas belle, arrangeons-la au moyen d'une fonction TABLE$. Ceci donnera deux colonnes propres en ajoutant des espaces et en ne conservant que la première partie du

résultat.

```
13100          PRINT TABLE$(NA$(Q),NC$(Q))

14000 DEFine FUNction TABLE$ (I1$,I2$)
14010   I1$=I1$ & FILL$(" ",16)
14020   I2$=I2$ & FILL$(" ",8)
14030   I1$=I1$(1 TO 16)
14040   I2$=I2$(1 TO 8)
14050     RETURN " "&I1$&I2$
14060 END DEFine TABLE$
```

NOM	CODE
ABRAHAM	A165
ABRAHAMS	A165
ABRAHAMS	A165
ABRAMS	A165
ADAM	A350
ADAMS	A352
ADDAMS	A352
ADAMSON	A352
ALAN	A450
ALLAN	A450
ALLEN	A450
ANTHANY	A535
ANTHONY	A535
ANTONY	A535
ANTROBUS	A536
APPERLEY	A164
APPLEBE	A141
APPLEBY	A141
APPLEFORD	A141

Il nous suffira de comparer le code du nom saisi avec les codes des DATAS.

```
  130     COMPARE

17000 DEFine PROCedure COMPARE

17030         FOR N=0 TO NW%
17040           IF CO$=NC$(N) THEN PRINT #2
                ,TABLE$(NA$(N),NC$(N))
17050         END FOR N

17090 END DEFine COMPARE
```

Ceci affichera seulement les mots qui ont le même code. Si on entre par exemple:

APPLEBEE A141

la réponse sera:

NOM	CODE
APPLEBEE	A141
APPLEBY	A141
APPLEFORD	A141

Bien que APPLEBE (un seul E) ne soit pas présent dans les DATAS, nous avons trouvé APPLEBEE, APPLEBY et APPLEFORD (la consonance finale a été éliminée).

EGALITE PARTIELLE

Remarquez que APPERLEY a été éliminé, bien que la consonnance soit proche. Il serait donc utile d'afficher des égalités partielles.

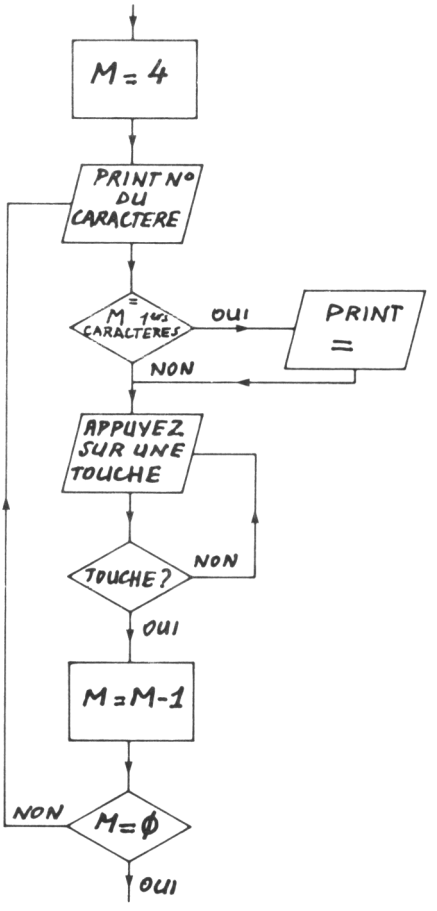
Ceci est facilement réalisable en ajoutant une boucle FOR NEXT qui compare une partie décroissante (1 à M) de l'entrée à avec des longueurs décroissantes des codes mémorisés (Organigramme 7.3).

```
17010   FOR M=4 TO 1 STEP -1
17020     PRINT #2,\,"comparaison sur les ";M;
          "premiers caractères"\

17040       IF CO$(1 TO M)=NC$(N)(1 TO M)
            THEN PRINT #2,TABLE$(NA$(N),NC$(N))

17060     PRINT #2,"appuyez sur une touche"
17070     DUMMY$=INKEY$(-1)
17080   END FOR M
```

Si on essaie maintenant on peut voir toutes les possibilités:

APPLEBEE A141

Comparaison sur les 4 premiers caractères:



Organigramme 7.3: Egalité partielle.

```
APPLEBEE    A141
APPLEBY     A141
APPLEFORD   A141

Appuyez sur une touche
```

Comparaison sur les 3 premiers caractères:

```
APPLEBEE     A141
APPLEBY      A141
APPLEFORD    A141

Appuyez sur une touche
```

Comparaison sur les 2 premiers caractères:

```
ABRAHAM      A165
ABRAHAMS     A165
ABRAMS       A165
APPERLEY     A164
APPLEBEE     A141
APPLEBY      A141
APPLEFORD    A141
```

Appuyez sur une touche

Comparaison sur les 1 premiers caractères:

```
ABRAHAM      A165
ABRAHAMS     A165
ABRAHAMS     A165
ABRAMS       A165
ADAM         A350
ADAMS        A352
ADDAMS       A352
ADAMSON      A352
ALAN         A450
ALLAN        A450
ALLEN        A450
ANTHANY      A535
ANTHONY      A535
ANTONY       A535
ANTROBUS     A536
APPERLEY     A164
APPLEBEE     A141
APPLEBY      A141
APPLEFORD    A141
```

Appuyez sur une touche

8

La reconnaissance des formes

Nous reconnaissons en général les objets avec nos sens, la vue, l'ouïe, le goût et le toucher, tandis que notre ordinateur ne peut obtenir des informations que par le moyen du clavier. Il est cependant possible de construire des détecteurs qui peuvent être interfacés avec notre machine, pour lui donner une autre perception du monde extérieur, mais cela suppose beaucoup de connaissances et d'habileté en électronique et en mécanique. A défaut nous allons simuler l'action d'un lecteur optique pour illustrer la reconnaissance des formes.

Pour commencer, imaginons trois formes simples, une ligne verticale, un carré et un triangle rectangle.

Nous reconnaissons facilement ces formes en regardant sur une grille fictive s'il y a un point ou non pour chaque couple de coordonnées X et Y.

Dans le cas de la ligne, on utilise une seule valeur de X et toutes celles de Y. Le carré est un peu plus compliqué, toutes les valeurs de X pour Y égale 1 ou 8, donnent un point, tandis que pour toutes celles de Y entre 2 et 7 il y a un point pour X égale 1 et 8. Pour le triangle c'est plus compliqué, la pente est obtenue en incrémentant X chaque fois.

Y	ligne	carré	triangle
1	1	255	1
2	1	129	3
3	1	129	5
4	1	129	9
5	1	129	17
6	1	129	33
7	1	129	65
8	1	255	255

Tableau 8.1: Valeur décimale des formes décrites en binaire.

Une façon évidente de décrire ces figures consisterait à représenter chaque point par un bit et calculer la valeur décimale de chaque rangée comme nous l'avons fait dans le chapitre sur les systèmes experts. (Tableau 8.1). De fait cette méthode est utilisée pour l'affichage des caractères à l'écran, leurs formats étant mémorisés de cette façon. La Figure 8.1 montre la construction de la lettre A.

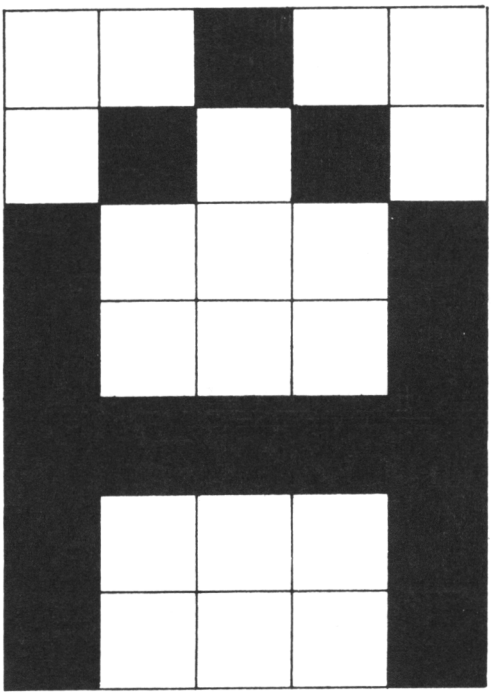


Figure 8.1: Dessin de la lecture A.

Il existe des machines (lecteurs optiques) capables de faire l'inverse et de lire une page imprimée en découpant l'espace selon une grille, et en mesurant la lumière réfléchie en des points particuliers.

Elles vont prendre en compte une série de "Oui" ou "Non" pour chaque coordonnée, puis décoder le résultat et le comparer à des formes connues. La façon la plus simple de faire cette comparaison est de considérer chaque point comme un bit puis de remplacer chaque rangée par son équivalent décimal qui peut être comparé à une table de valeurs connues. Il y a cependant un inconvénient, il faut comparer tous les points de la grille, 64 ici.

UN ELAGAGE.

Une approche plus rapide s'appuie sur le fait que chaque caractère peut être reconnu en examinant un nombre de points critiques beaucoup plus petit. Par exemple, la Figure 8.2 donne l'arbre de décision qui permettra de reconnaître toutes les majuscules à partir de douze points seulement, de plus il n'est pas nécessaire d'en tester douze dans tous les cas. (Figure 8.3). Si vous suivez tous les branches, vous verrez que le nombre maximum est de 7 étapes, et que la plupart des lettres sont trouvées en moins de 5. (Tableau 8.2). C'est quand même plus rapide que d'en tester 64!

Pour montrer le fonctionnement, nous allons simuler l'action de la tête de lecture au moyen d'une grille sur laquelle nous allons construire les caractères.

La routine INITIALISATION fait le nettoyage initial pour préparer l'affichage.

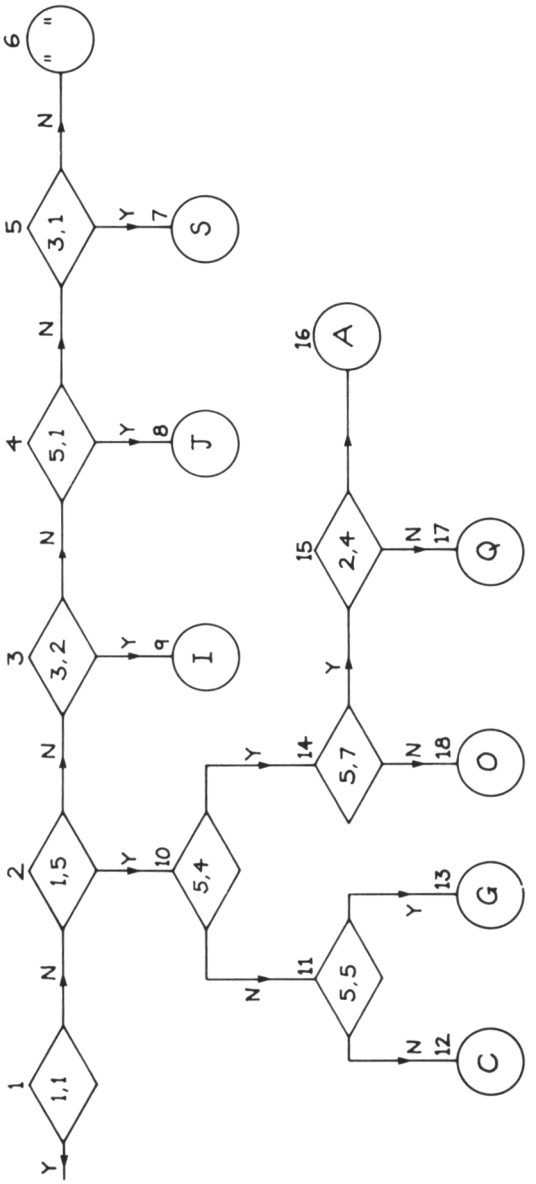


Figure 8.2(a): Arbre de décision pour l'alphabet.

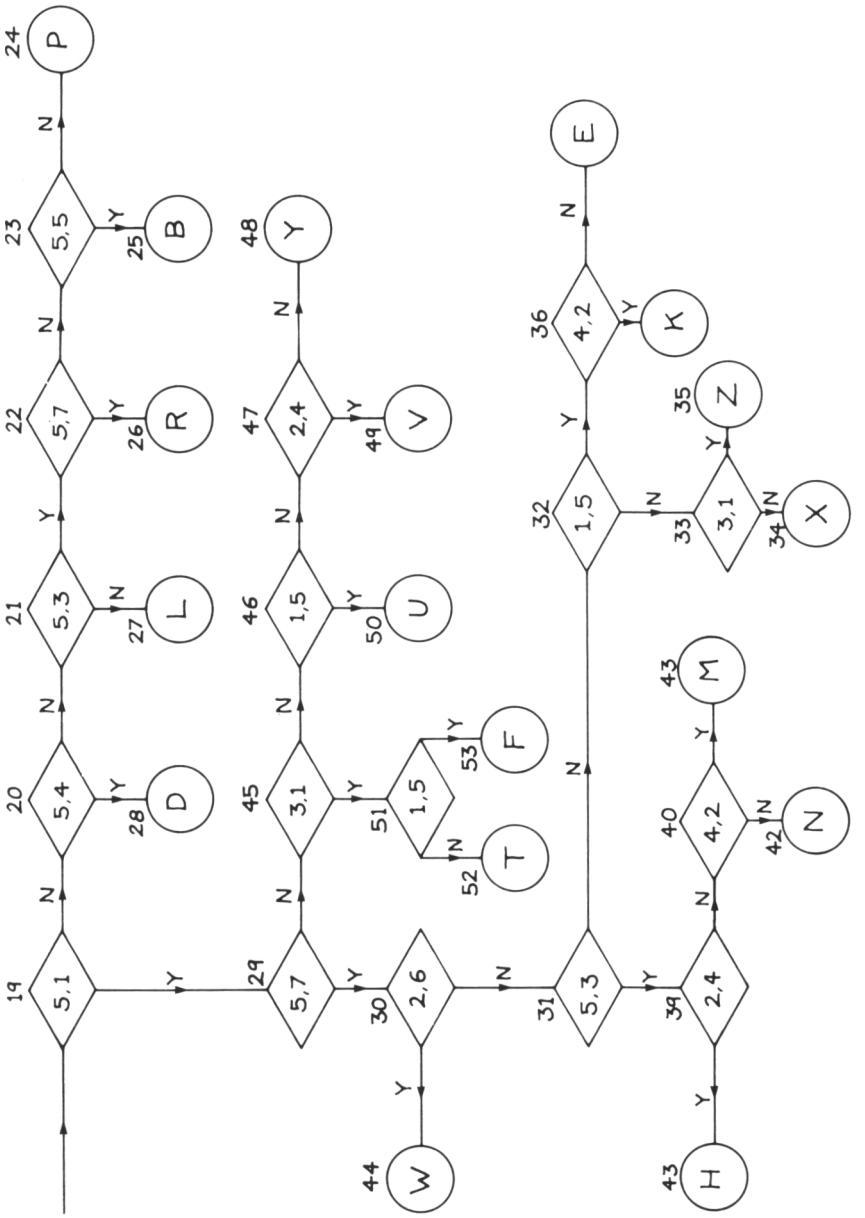

19
5,1
20
5,4
28
D
21
5,3
27
L
22
5,7
26
R
23
5,5
25
B
24
P
29
5,7
30
2,6
44
W
45
3,1
51
1,5
53
F
52
T
46
1,5
50
U
47
2,4
49
V
48
Y
31
5,3
32
1,5
33
3,1
34
X
35
Z
36
4,2
K
E
39
2,4
43
H
40
4,2
42
N
43
M
Y
N

Figure 8.2(b)

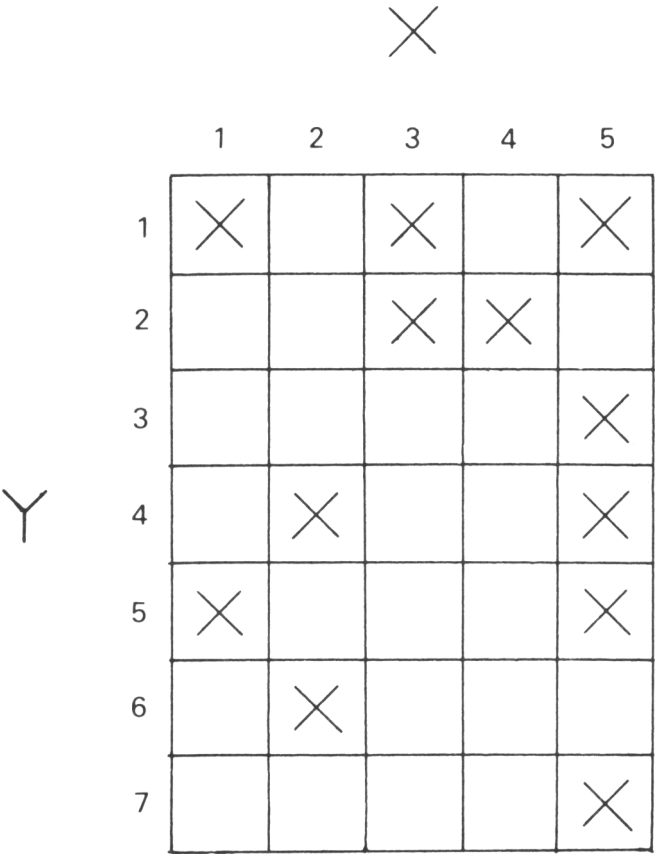


Figure 8.3: Points utilisés.

3 étapes	:	I D
4 "	:	L J C G O W
5 "	:	S A Q R T F U espace
6 "	:	P V Y H
7 "	:	B M N E K X Z

Tableau 8.2: Nombre d'étapes pour la reconnaissance des caractères.

```
   10 INITIALISATION

11000 DEFine PROCedure ECRAN
11010   MODE 4
11020   PAPER 0
11030   CLS
11040   WINDOW #2,270,200
11050   BORDER #2,3,7
11060   CSIZE #2,1,0
11070   CLS #2
11080   WINDOW #1,144,200,318,16
11090   BORDER #1,1,7
11100   CSIZE #1,1,0
11110   PAPER #1,5
11120   INK #1,0
11130   CLS #1
11140   BORDER #0,10,0
11150   INK #0,7
11160   CLS #0
11170 END DEFine ECRAN
```

Le tableau de décision est décrit dans une suite de tables chaînées où NB est le nombre de branches, LE$(N) contient le nom des lettres, C1(N) la prochaine valeur de X à tester, C2(N) la prochaine valeur de Y, N(N) l'élément suivant dans le cas d'une réponse négative, et Y(N) l'élément suivant dans le cas d'une réponse positive.

```
10020     AT #2,5,10 : PRINT #2,"CHARGEMENT DES
          DATAS"
10030     ARBRE
10040     CLS #2

12000 DEFine PROCedure ARBRE
12010   RESTORE
```

```
12020   NB=53 : DIM LE$(NB),C1(NB),C2(NB)
        ,N(NB),Y(NB)
12030     FOR C=1 TO NB
12040       AT #2,15,12 : PRINT #2,C
12050       READ LE$(C) : READ C1(C) :
            READ C2(C) : READ N(C) : READ Y(C)
12060     END FOR C
12070   DATA 0,0,0,2,19
12080   DATA 0,0,4,3,10
12090   DATA 0,2,1,4,9
12100   DATA 0,4,0,5,8
12110   DATA 0,2,0,6,7
12120   DATA "_",0,0,0,0
12130   DATA "S",0,0,0,0
12140   DATA "J",0,0,0,0
12150   DATA "I",0,0,0,0
12160   DATA 0,4,3,11,14
12170   DATA 0,4,4,12,13
12180   DATA "C",0,0,0,0
12190   DATA "G",0,0,0,0
12200   DATA 0,4,6,10,15
12210   DATA 0,1,3,17,16
12220   DATA "A",0,0,0,0
12230   DATA "Q",0,0,0,0
12240   DATA "O",0,0,0,0
12250   DATA 0,4,0,20,29
12260   DATA 0,4,3,21,28
12270   DATA 0,4,2,27,22
12280   DATA 0,4,6,23,26
12290   DATA 0,4,4,24,25
12300   DATA "P",0,0,0,0
12310   DATA "B",0,0,0,0
12320   DATA "R",0,0,0,0
12330   DATA "L",0,0,0,0
12340   DATA "D",0,0,0,0
12350   DATA 0,4,6,45,30
12360   DATA 0,1,5,31,44
12370   DATA 0,4,2,32,39
12380   DATA 0,0,4,33,36
12390   DATA 0,2,0,34,35
12400   DATA "X",0,0,0,0
12410   DATA "Z",0,0,0,0
12420   DATA 0,3,1,38,37
12430   DATA "K",0,0,0,0
12440   DATA "E",0,0,0,0
12450   DATA 0,1,3,40,43
12460   DATA 0,3,1,42,41
```

```
12470   DATA "M",0,0,0,0
12480   DATA "N",0,0,0,0
12490   DATA "H",0,0,0,0
12500   DATA "W",0,0,0,0
12510   DATA 0,2,0,46,51
12520   DATA 0,0,4,47,50
12530   DATA 0,1,3,48,49
12540   DATA "Y",0,0,0,0
12550   DATA "V",0,0,0,0
12560   DATA "U",0,0,0,0
12570   DATA 0,0,4,52,53
12580   DATA "T",0,0,0,0
12590   DATA "F",0,0,0,0
12600 END DEFine ARBRE
```

Un tableau GRILLE 5 X 7 est défini pour contenir les points du caractère à produire (attention aux éléments nuls), et le curseur est placé en haut à gauche (X%=0, Y%=0).

```
10050   DIM GRILLE(4,6)
10060   X%=0 : Y%=0
```

Les fonctions des touches sont indiquées dans la partie gauche de l'écran et nous sommes prêts à utiliser la procédure EDITEUR pour dessiner notre caractère.

```
20   REPeat CARACTERE
30     PRINT #2,\"BARRE ESPACEMENT PLACE
       UN POINT"
40     PRINT #2,\"F1 EFFACE UN POINT"
50     PRINT #2,\"F2 EFFACE L'ECRAN"
60     PRINT #2,\"F3 DECODE"
70     EDITEUR

90  END REPeat CARACTERE
```

Une représentation du contenu du tableau GRILLE, avec un curseur clignotant, est affichée dans la fenêtre de droite (#1) par l'EDITEUR. Une boucle met à la couleur 2 (rouge) le pavé courant (X%,Y%), puis regarde si l'élément correspondant de GRILLE contient 1 (GRILLE(X%,Y%) est alors vrai). Si on trouve 1, le pavé est mis à la couleur 0 (noir). Dans le cas contraire, le pavé est colorié en vert (0), pour éviter un ef-

fet de rémanence. Le clignotement dépend du délai de INKEY$(N), et la séquence se répète jusqu'à ce qu'on appuie sur une touche.

```
13000 DEFine PROCedure EDITEUR
13010   REPeat LOOP
13020     BLOCK 28,28,(X%*28),(Y%*28),2
13030       IF GRID(X%,Y%) THEN
13040         BLOCK 28,28,(X%*28),(Y%*28),0
13050           ELSE
13060             BLOCK 28,28,(X%*28),(Y%*28),5
13070       END IF
13080     A$=INKEY$(5)
13090      IF A$="" THEN END REPeat LOOP
```

Lorsque l'on appuie sur une touche, le code de celle-ci est récupéré puis soumis à une série de tests. Les coordonnées sont mises à jour suivant les touches de direction, et l'appui sur la barre d'espacement colore le pavé en noir et l'élément correspondant du tableau GRILLE est mis à 1. En cas d'erreur la touche F1 efface la position courante, la remet en vert et place 0 dans l'élément de GRILLE. Remarquez qu'il faut ajouter des tests pour éviter de sortir des limites de la grille.

```
13100   A=CODE(A$)
13110     IF A=192 AND X%>0 THEN X%=X%-1 :
          END REPeat LOOP
13120     IF A=200 AND X%<4 THEN X%=X%+1 :
          END REPeat LOOP
13130     IF A=208 AND Y%>0 THEN Y%=Y%-1 :
          END REPeat LOOP
13140     IF A=216 AND Y%<6 THEN Y%=Y%+1 :
          END REPeat LOOP
13150     IF A=32 THEN GRILLE(X%,Y%)=1 :
          BLOCK 28,28,(X%*28),(Y%*28),0 :
          END REPeat LOOP
13160     IF A=232 THEN GRILLE(X%,Y%)=0 :
          BLOCK 28,28,(X%*28),(Y%*28),4 :
          END REPeat LOOP
```

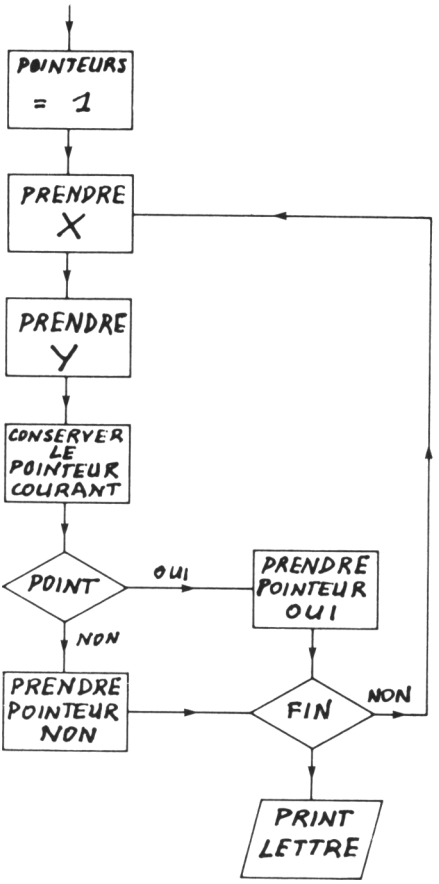
Si le dessin de votre caractère se termine par une déconfiture complète, la touche de fonction F2 efface toute la fenêtre 1, et réinitialise la GRILLE à 0.

```
13170      IF A=236 THEN
13180        CLS #1
13180          FOR C=0 TO 4
13190            FOR M=0 TO 6
13200              GRILLE(C,M)=0
13210            END FOR M
13220          END FOR C
13240    END REPeat LOOP
13250      END IF
```

Enfin la touche de fonction F3 renvoie à la PROCedure LECTEUR qui décode le dessin ou bien le programme recommence l'analyse des touches.

```
13260          IF A=240 THEN RETurn
13270    END REPeat LOOP
13280 END DEFine EDITEUR

   80      LECTEUR
```

Dans la PROCedure LECTEUR, le dessin réalisé est comparé aux figures connues (Organigramme 8.1). Le tableau de pointeurs AP, est initialisé à 1, de façon à commencer la recherche au début. Les coordonnées X et Y sont lues dans C1(AP%) et C2(AP%), et la dernière position LP% prend la valeur du pointeur courant AP%. La couleur du point PC pour ces coordonnées est déterminée par l'élément correspondant de GRILLE. Si celui-ci contient 1, il y a un point et il faut prendre le pointeur 'oui' Y(AP%). S'il ne contient pas 1, il faut prendre le pointeur 'non'. Dans les deux cas il faut voir si l'élément pointé contient 0 (extrémité de la branche), ce qui indique qu'un caractère a été trouvé. La lettre LE$(LP%) est affichée dans la fenêtre # 0, jusqu'à ce que l'on appuie sur une autre touche, ce qui initialise un nouveau cycle. Tant qu'une valeur supérieure à 0 est trouvée, il y a un nouveau point sur la branche et le programme recommence avec les nouvelles valeurs C1(AP%) et C2(AP%). Pour vous permettre de voir les points qui ont été testés, ces pavés sont coloriés en rouge. Les points marqués mais non testés restent en noir.



Organigramme 8.1: PROCedure lecteur.

```
14000 DEFine PROCedure LECTEUR
14010   AP%=1
14020     REPeat PIXEL_CHECK
14030       X%=C1(AP%) : Y%=C2(AP%) : LP%=AP%
14040       PC%=GRILLE(X%,Y%)
14060         IF PC% THEN AP%=Y(AP%) :
              ELSE AP%=N(AP%)

14080         IF AP% THEN
14090           BLOCK 28,28,(X%*28),(Y%*28),2
14100     END REPeat PIXEL_CHECK
14110         END IF
14120      CSIZE #0,3,1 : AT #0,6,0 :
           PRINT #0,"CARACTERE : ";LE$(LP%); :
           CSIZE #0,0,0
14130      PRINT #2,\\"APPUYEZ SUR UNE CLE"
```

Pour voir les branches de l'arbre qui sont parcourues, apportez les modifications suivantes.

```
14140      A$=INKEY$(-1)
14150      CLS #0 : CLS #1 : CLS #2
14160 END DEFine LECTEUR
```

L'inconvénient de cette méthode rapide des points critiques, est que l'on obtiendra une erreur de comparaison dans le cas où la forme n'est pas prévue dans l'arbre. Les premiers lecteurs optiques ne supportaient qu'un seul type de caractères, mais actuellement non seulement ils acceptent des styles différents, mais encore ils apprennent par eux-mêmes les règles de reconnaissance au moyen d'un système expert interne. On leur apprend en leur montrant quelques pages de texte, tout en les rentrant au clavier. Cependant on est encore loin de pouvoir lire l'écriture manuscrite.

9

Un professeur intelligent

L'enseignement assisté par ordinateur (E.A.O.) est un autre domaine où l'intelligence artificielle peut se montrer particulièrement utile. C'est très bien d'avoir un programme qui contrôle de façon aléatoire les connaissances d'un étudiant, mais ce n'est pas comme cela que travaillent les professeurs. Tout en posant des questions ils gardent un oeil sur les progrès des élèves et augmentent la difficulté des questions avec l'expérience, puis les interrogent plus précisement sur les types de problèmes pour lesquels ils rencontrent des difficultés. Par exemple, si un enfant subit un contrôle sur les quatre opérations, et que seules les réponses sur la division comportent des erreurs, il faudra pour être plus efficace lui poser plus de questions sur la division;

Voyons comment on peut introduire cet aspect 'humain' dans un programme.

QUESTIONS ET REPONSES

Nous allons utiliser des nombres aléatoires pour la première question, l'addition par exemple. RND(0 TO 10) nous donnera des nombres entre 0 et 10.

```
10 ECRAN
20 REPeat QUESTION
40     A%=RND(0 TO 10)
50     B%=RND(0 TO 10)
```

```
10000 DEFine PROCedure ECRAN
10010    MODE 4
10020    CLS#0 :CLS#1 : CLS#2
10030 END DEFine ECRAN
```

L'ordinateur fait la somme, puis appelle la PROCedure VERIF.

```
60    C%=C%+B%
70    VERIF
```

VERIF imprime la question, puis saisit la réponse IP%.

```
1000 DEFine PROCedure VERIF
1010   PRINT\ A%;"+";B%;"=";
1020   INPUT IP%
```

Votre réponse doit être contrôlée. Si c'est la bonne réponse, C%, on imprime "JUSTE", sinon "FAUX" suivi de la réponse, puis on pose la question suivante.

```
1030   IF C%=IP% THEN
1040     PRINT\"JUSTE"
1050    ELSE PRINT\"FAUX, LA BONNE REPONSE ES
T ";C%
1060   END IF
1080 END DEFine VERIF

280    END REPeat QUESTION
```

SI CE N'EST PAS UN NOMBRE?

Un essai vous montrera que la machine se "plante" si vous rentrez une lettre à la place d'un chiffre, accidentellement ou non. Il serait bien plus agréable que le professeur refuse d'accepter autre chose qu'un nombre, aussi allons-nous utiliser une PROCedure de saisie au lieu du simple INPUT. Celle-ci saisit une chaîne IP$, plutôt qu'un nombre, regarde si elle est vide et boucle dans ce cas. Elle vérifie que le code de chaque caractère de la chaîne IP$(N) est numérique (codes de 42 à 56),

avant de la convertir en une simple variable IP%.

```
1020   SAISIS
2000 DEFine PROCedure SAISIS
2010   INPUT IP$
2020    IF IP$="" THEN SAISIS
2030      FOR N=1 TO LEN(IP$)
2040      IF CODE (IP$(N))<43 OR CODE (IP$(N)
)>57 THEN
2050        PRINT "ENTREZ UN NOMBRE !!  ";
2060        SAISIS
2070      END IF
2080      END FOR N
2090    IP%=IP$
2100 END DEFine SAISIS
```

D'AUTRES OPERATIONS

Les trois autres opérations peuvent être facilement traitées de la même façon en remplaçant le signe '+' par une chaîne SG$ que nous initialiserons dans chaque cas avec le bon opérateur. En même temps,comme RND(0 TO 10) est commun à tous les calculs, nous le mettrons dans une fonction appelée PICK qui retourne un nombre.

```
1010   PRINT\A%;SG$;B%;"=";

3000 DEFine FuNction PICK
3020     RETurn RND(0 TO 10)
3030 END DEFine PICK

 40      A%=PICK
 50      B%=PICK
 60      C%=A%+B%:SG$="+"
 70      VERIF
 100     A%=PICK
 110     B%=PICK
 120     C%=A%-B%:SG$="-"
 130     VERIF
 160     A%=PICK
 170     B%=PICK
 180     C%=A%*B%:SG$="*"
```

```
190     VERIF
220     A%=PICK
230     B%=PICK
240     C%=A%/B%:SG$="/"
250     VERIF
```

DIVISION PAR ZERO

Ecrit ainsi, le programme se plante si B% vaut 0 et si le choix se porte sur la division. On peut l'éviter en ajoutant systématiquement 1 à B%:

```
230       B%=PICK+1
```

ELIMINATION DES DECIMALES

Nous utilisons des variables entières pour avoir des nombres ronds, mais il est évident qu'une division peut donner un nombre fractionnaire, que l'on ne peut pas saisir correctement car IP% sera arrondi: 3/2 = 1.5.

Mais le programme acceptera 1, 1.5, 1.9 et tout nombre entre 1 et 1.999.

Pour éviter les décimales, il faut que A% soit un multiple de B%. Nous calculons donc B% d'abord, et nous prenons A% égal à B% multiplié par un nombre aléatoire entre 0 et 10.

```
220       B%=PICK
230       A%=PICK*B%
```

MEMORISATION DU RESULTAT

Maintenant que le contrôle marche bien il faut réfléchir à la façon de garder les résultats de l'élève. La méthode la plus simple est d'incrémenter une variable TR%, à chaque appel de la PROCedure SAISIS, et une variable

résultat chaque fois que la réponse est bonne.

```
  10 ECRAN: TR%=0 : SC%=0

1040      PRINT \"JUSTE" :SC%=SC%+1

2090      IP%=IP$:TR%=TR%+1
```

On peut consulter les résultats avec la PROCedure SCORE appelée à la fin de VERIF.

```
1070      SCORE

4000 DEFine PROCedure SCORE
4010   PRINT "VOTRE RESULTAT EST ";SC%;"/";TR%
4020 END DEFine SCORE
```

Si vous préférez le résultat en pourcentage modifiez ainsi la ligne 4010:

```
4010   PRINT "VOUS AVEZ EU ";(SC%/TR%)*100;" %
       DE REPONSES JUSTES"
```

COMBIEN DE QUESTIONS?

Le programme pose successivement une question de chaque type, à l'infini. On peut s'arrêter en mettant le nombre de questions dans une variable NQ%.

```
10 ECRAN: TR%=0 : SC%=0 : NQ=32
```

A chaque question NQ% est décrémenté de 1, quand il vaut 0 on s'arrête. (après quatre opérations de chaque type).

```
2090      IP%=IP$:TR%=TR%+1:NQ%=NQ%-1

3010      IF NQ%=0 THEN PRINT "32 QUESTIONS
          POSEES"
```

SUIVI DE LA PROGRESSION.

Si nous voulons orienter les questions selon la difficulté, il faut garder le résultat de chaque opération. Il nous faut des variables séparées pour chaque type de question (AD% pour l'addition, SU% pour la soustraction, MU% pour la mutiplication, et DI% pour la division). Ces variables sont définies comme le huitième du nombre total de questions à poser, NQ%.

```
10 ECRAN:TR%=0:SC%=0:NQ%=32:AD%=NQ%/32:SU%=NQ
%/32:MU%=NQ%/32:DI%=NQ%/32
```

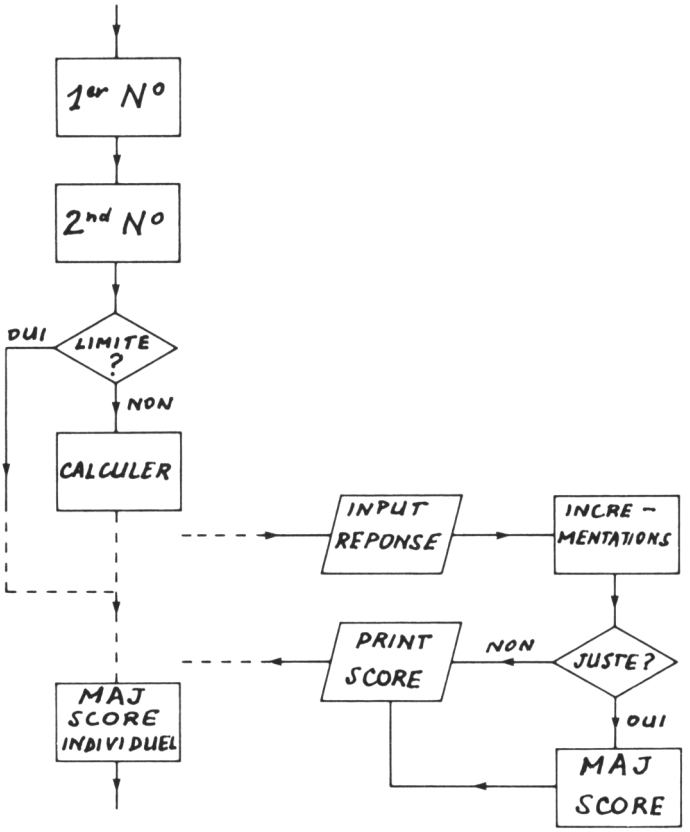
Maintenant si la bonne réponse C% est la même que la vôtre, IP% un compteur IN% est mis à -1, "JUSTE" est affiché, et on sort. Sinon IN% est mis à 1, "FAUX" est affiché suivi de la bonne réponse.

```
1040      PRINT\"JUSTE":IN%=-1
1050    ELSE PRINT\"FAUX, LA BONNE REPONSE ES
T ";C%:IN%=1
```

IN% est ajouté au bon nombre d'opérations (AD%, SU%, MU% ou DI%) en sortant de VERIF, donc si la réponse est mauvaise IN% augmente et inversement.

```
70     VERIF : AD%=AD%+IN%

130    VERIF : SU%=SU%+IN%

190    VERIF : MU%=MU%+IN%

250    VERIF : DI%=DI%+IN%
```

Ajoutons un test pour voir si toutes les questions d'un type particulier ont reçu une bonne réponse (AD%=<0, organigramme 9.1). Dans ce cas on n'en posera plus en sautant par-dessus ce type d'opération.



Organigramme 9.1: Un professeur intelligent.

```
 30        IF AD%>0 THEN

 80        END IF
 90        IF SU%>0 THEN

140        END IF
150        IF MU%>0 THEN

200        END IF
210        IF DI%>0 THEN

260        END IF
```

Si les questions de chaque type ont reçu une bonne réponse, le programme s'arrête (AD%=0,

```
270   IF AD%+SU%+MU%+DI%=0 THEN PRINT
      "4 QUESTIONS DE CHAQUE TYPE JUSTES"
```

Remarquez que l'on ne pose plus de question dans les domaines où il y a eu quatre réponses sans erreur. Si vous faites une erreur, AD% (etc.) sera incrémenté, il faudra donc plus de bonnes réponses pour que AD% atteigne 0.

DEGRE DE DIFFICULTE

Comment rendre les questions plus faciles ou plus difficiles selon les réponses, c'est-à-dire modifier les valeurs de AD%, SU%, MU%, DI%? Jusqu'à présent les valeurs de A% et B% sont comprises entre 0 et 10; nous allons modifier ces nombres pour obtenir des valeurs plus fortes si les réponses sont bonnes et inversement. En même temps on s'assure que les valeurs produites ne soient pas négatives quand les réponses sont particulièrement mauvaises.

Modifions tout d'abord la FONCtion PICK de façon à ce que le compteur X% contienne le nombre de questions de chaque type à poser.

```
 40       A%=PICK(AD%)
 50       B%=PICK(AD%)

100       A%=PICK(SU%)
110       B%=PICK(SU%)

160       A%=PICK(MU%)
170       B%=PICK(MU%)

220
230

3000 DEFine FuNction PICK(X%)
```

Le cas le plus défavorable est celui où toutes les réponses sont fausses dans le dernier groupe. Dans ce cas il y aura seulement quatre questions dans les trois premiers et 32-(3*4)=20 dans le dernier. De plus il faut se souvenir

que X% (c'est-à-dire AD%) commence à 4, le maximum de X% obtenu est donc de 20+4=24.

Nous rajoutons donc une variable de pondération, WT% calculée en soustrayant trois fois le nombre de questions à poser dans chaque groupe (3*AD%) du nombre total des questions (NQ%), puis en rajoutant la valeur initiale de AD%.

```
WT%=NQ%-(3*AD%)+AD%
```

soit plus simplement:

```
XT%=NQ%-(2*AD%)
```

```
10 ECRAN: TR%=0 : SC%=0 : NQ%=32 :
   AD%=NQ%/32 : SU%=NQ%/32 : MU%=NQ%/32 :
   DI%=NQ%/32 : WT%=NQ%-(2*AD%)
```

Nous remplaçons maintenant la constante 10 par la différence entre WT% et X%, en modifiant le calcul dans la fonction PICK.

```
3020    RETurn RND(0 TO (WT%-X%))
```

Initialisons WT% à 24 et X% à 4, de façon à obtenir des nombres entre 0 et 20. Dans le cas d'une bonne réponse X% est réduit à 3, et des nombres entre 0 et 21 seront choisis. Après quatre bonnes réponses, X% atteint 0 et ne changera pas pour ce type de question car la ligne sera sautée. Les dernières valeurs seront alors comprises entre 0 et 23.

Si la première réponse est fausse, X% sera incrémenté de 1, et les valeurs choisies iront de 0 à 19. Dans le cas le plus défavorable, X% sera incrémenté 20 fois, pour atteindre 24 et WT%-X% vaudra 0 pour A% et B%. (A vous de résoudre ce problème particulier).

D'AUTRES TYPES DE QUESTIONS

Bien que l'exemple ci-dessus ne traite que de problèmes arithmétiques, la même technique peut être utilisée sans difficulté particulière dans d'autres domaines, en utilisant des questions et des réponses composées de phrases.

10

Des souris et des hommes

Depuis des siècles les hommes sont fascinés par les labyrinthes, et les difficultés rencontrées pour trouver la sortie ont été abondamment décrites dans la mythologie. Plus récemment le sujet a été repris par la bande enthousiaste des 'mousquetaires' qui envoient leur souris électronique se battre contre l'inconnu. Certains doivent penser que ce sont des jeux futiles, mais s'il s'agissait d'envoyer une personne contrôler le rayonnement après une explosion nucléaire, ou explorer la surface d'une planète étrangère, ils leur laisseraient volontiers la place. Un guidage direct à courte distance est possible, associé à une caméra vidéo, il peut permettre à un opérateur de trouver son chemin, mais les délais impliqués par les transmissions à grande distance posent des problèmes considérables. Cela ne vous avance pas à grand'chose de voir que votre véhicule d'exploration, qui a coûté des milliards, va tomber dans une crevasse de la planète Mars, si la chute a déjà eu lieu au moment où vous recevez l'image! Des machines autonomes et intelligentes auront donc toujours leur place. Un robot explorateur sera équipé de censeurs adaptés à son environnement et à son activité, il devra avoir un moteur fiable. Avec notre QL, nous pouvons seulement simuler quelques problèmes de cheminement.

PLACONS LE DECOR

Pour commencer initialisons un écran avec trois fenêtres. A droite (#1), nous montrons le

labyrinthe, à gauche (#2) nous affichons le contenu du cerveau de notre souris, et en bas (#0) nous avons l'heure et l'état.

```
10 ECRAN
10000 DEFine PROCedure ECRAN
10010   MODE 4
10020   WINDOW #2,230,200,25,15
10030   BORDER #2,1,4
10040   PAPER #2,7
10050   INK #2,0
10060   CSIZE #2,1,0
10070   WINDOW #1,230,200,258,15
10090   BORDER #1,1,4
10100   PAPER #1,0
10110   INK #1,7
10120   CSIZE #1,1,0
10130   CLS #1
10140   PAPER #0,0
10150   INK #0,0
10160   CSIZE #0,2,0
10170   CLS #0
10180   PAPER #0,7
10190   PRINT #0,"    VU PAR LA SOURIS      "
10200 END DEFine ECRAN
```

CONSTRUCTION DU LABYRINTHE

Il faut construire le labyrinthe que nous allons explorer. Nous pourrions le générer au hasard, mais il est plus amusant de le dessiner soi-même, et cela rend plus faciles les essais pour éclaircir les situations confuses. Le labyrinthe est contenu dans un tableau de 37 sur 33, et une copie des éléments est affichée dans la fenêtre #1. Ici chaque élément est représenté par un bloc de 6 X 6 pixels. Avant de partir montrons en vert (couleur 4) le centre du labyrinthe (18,16). La position de départ est le coin en haut à gauche (X%=&, Y%=1).

```
20 DESSIN
```

```
11000 DEFine PROCedure DESSIN
11010   DIM MAZE(37,33)
11020   X%=1 : Y%=1
11030   BLOCK 6,6,(18*6),(16*6),4
```

Tant qu'une touche n'est pas enfoncée, on boucle en promenant un curseur tantôt vert (couleur 4), tantôt de la couleur de la case courante.

```
11040   REPeat BOUCLE
11050     BLOCK 6,6,X%*6,Y%*6,4
11060     BLOCK 6,6,X%*6,Y%*6,MAZE(X%,Y%)
11070       A$=INKEY$(2)
11080         IF A$="" THEN END REPeat BOUCLE
```

Quand une touche est enfoncée, on teste les quatre touches de direction. Tant que l'on reste dans les limites du tableau, les coordonnées X et Y sont mises à jour, et le curseur se déplace (sans laisser de trace).

```
11090             A=CODE(A$)
11100               IF A=192 AND X%>1 THEN X%
=X%-1:END REPeat BOUCLE
11130               IF A=200 AND X%<35 THEN X
%=X%+1:END REPeat BOUCLE
11160               IF A=208 AND Y%>1 THEN Y%
=Y%-1:END REPeat BOUCLE
11190               IF A=216 AND Y%<31 THEN Y
%=Y%+1:END REPeat BOUCLE
```

En dessinant le labyrinthe il faudra marquer le chemin à suivre par la souris. Nous le montrerons par des pavés blancs sur l'écran (couleur 6). Il sera ainsi facile de modifier le labyrinthe en changeant la couleur des pavés, avec la procédure CHEMIN qui contient un paramètre pour la couleur à utiliser. Attention on doit mettre à jour le dessin et l'affichage.

```
12000 DEFine PROCedure CHEMIN (COULEUR)
12010     MAZE(X%,Y%)=COULEUR
12020     BLOCK 6,6,X%*6,Y%*6,COULEUR
12030 END DEFine CHEMIN
```

La touche CTRL associée à une touche de direction permet de créer des pavés blancs, tandis que ALT avec une touche de direction donne un pavé noir.

```
11110          IF A=193 AND X%>1 THEN CHEMIN 0
:X%=X%-1:END REPeat BOUCLE
11120          IF A=194 AND X%>1 THEN CHEMIN 6
:X%=X%-1:END REPeat BOUCLE
11140          IF A=201 AND X%<35 THEN CHEMIN
0:X%=X%+1:END REPeat BOUCLE
11150          IF A=202 AND X%<35 THEN CHEMIN
6:X%=X%+1:END REPeat BOUCLE
11170          IF A=209 AND Y%>1 THEN CHEMIN 0
:Y%=Y%-1:END REPeat BOUCLE
11180          IF A=210 AND Y%>1 THEN CHEMIN 6
:Y%=Y%-1:END REPeat BOUCLE
11200          IF A=217 AND Y%<31 THEN CHEMIN
0:Y%=Y%+1:END REPeat BOUCLE
11210          IF A=218 AND Y%<31 THEN CHEMIN
6:Y%=Y%+1:END REPeat BOUCLE
```

Si votre dessin de labyrinthe se présente mal, SHIFT F1 permettra de relancer le programme.

```
11220          IF A=234 THEN RUN
```

Enfin SHIFT F2 fera sortir de la PROCedure DESSIN, et la souris commencera sa recherche. Remarquez que le labyrinthe ne doit répondre qu'à une condition: il doit y avoir un chemin quelconque entre le point de départ (X%=1, Y%=1) et le centre.

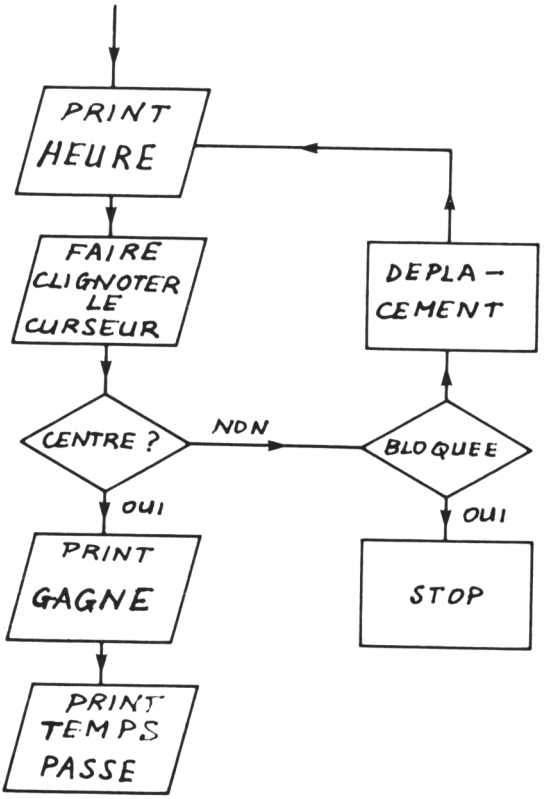
```
11230          IF A=238 THEN  RETurn
11240     END REPeat BOUCLE
11250 END DEFine DESSIN
```

TROUVER LE CHEMIN

Nous pouvons maintenant envoyer la souris à la recherche du fromage au centre du labyrinthe. Il faut lui donner une mémoire de la taille de la grille, une position de départ, et initiali-

ser l'heure avec SDATE, pour chronomètre ses progrès.

```
30   REPeat RESTART

40      DIM MEMORY(37,33) : X%=1 : Y%=1

60      SDATE 1985,0,0,0,0,0
```



Organigramme 10.1: Déplacement.

L'organigramme 10.1 montre la boucle de déplacement de la souris. Affichons tout d'abord les cinq derniers caractères de DATE$, c'est-à-dire les minutes et les secondes. Notez que l'on ne peut pas découper DATE$ lui-même, il faut d'abord le convertir en une variable temporaire D$.

```
70      REPeat MOUVEMENT
80        D$=DATE$
```

```
90        D$=D$(16 TO)
100       AT #0,16,0 : PRINT #0,D$

180    END REPeat MOUVEMENT
```

Le point de départ est mis à vert (couleur 4) dans MEMORY, et un curseur clignotant est affiché par la PROCedure TRACE à la position correspondante dans la fenêtre #2. Ceci nécessite trois paramètres, X et Y pour les coordonnées, et la couleur.

```
110      MEMORY(X%,Y%)=4
120      TRACE X%,Y%,0
130      TRACE X%,Y%,MEMORY(X%,Y%)

1000 DEFine PROCedure TRACE (X1,Y1,C)
1010     BLOCK #2,6,6,(X1*6),(Y1*6),C
1020 END DEFine TRACE
```

EST-CE GAGNÉ?

On peut facilement voir si le centre a été atteint en testant les coordonnées (18,16). Quand le centre est atteint, si on y arrive, la durée du déplacement est affichée, et on a trois options: N permet de dessiner un nouveau labyrinthe; C efface l'écran et permet un nouvel essai avec le même dessin, seul le tableau MEMORY est réinitialisé. Toute autre touche fait repartir sur le même labyrinthe sans effacer l'écran, ce qui permet de comparer les différents trajets.

```
140       IF X%=18 AND Y%=16 THEN CENTRE


2000 DEFine PROCedure CENTRE
2010     PRINT #0,"LE CENTRE EST ATTEINT "\" E
N ";D$(1 TO 2);" MINUTES ET ";D$(4 TO 5);" SE
CONDES"
2020     A$=INKEY$(-1)
2030       IF A$="N" THEN RUN
2040       IF A$="C" THEN CLS #2:END REPeat RE
START
2050     END REPeat RESTART
2060 END DEFine CENTRE
```

```
2060 END DEFine CENTRE
```

QUEL CHEMIN?

La souris doit d'abord regarder où elle peut se déplacer. La position suivante à examiner est obtenue en additionnant aux coordonnées courantes celles du déplacement XD% et YD%. Pour commencer on initialisera XD% à 0 et YD% à 1, de façon à ce que la souris essaye de descendre.

```
 50     XD%=0:YD%=1

150          IF MAZE(X%+XD%,Y%+YD%)=0 THEN STOP

170        X%=X%+XD%:Y%=Y%+YD%
```

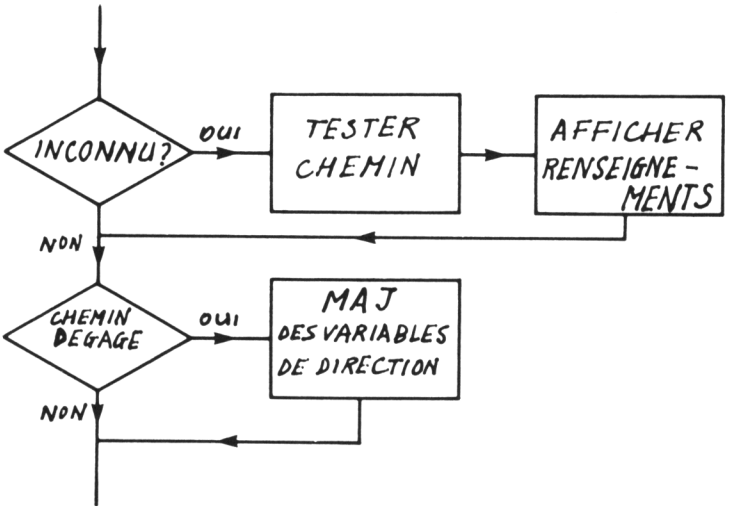
Lancez le programme, dessinez une simple ligne verticale, et lâchez la souris en appuyant sur SHIFT F2. Vous la verrez descendre jusqu'à l'extrémité de la ligne et s'arrêter. Il faut maintenant envisager ce qu'il faut faire lorsque la position suivante ne contient pas 0.

LES COINS

La première idée qui vous vient à l'esprit est de changer de direction si elle atteint un mur (inverser XD% et YD% en les multipliant par -1), mais cela conduira la souris à se cogner sans fin sur le mur. Si elle est capable de prendre une nouvelle direction, elle peut explorer son environnement. Quatre fonctions sont définies pour les quatre directions possibles, gauche, droite, haut et bas. Chacune travaille de la même façon, en utilisant une PROCedure INCONNU pour examiner la position suivante. Jusqu'ici nous avons examiné le labyrinthe, sans conserver d'information en mémoire. Tant que l'on a pas d'information dans une position de MEMORY celle-ci reste à 0. Copions les informations dans MEMORY et faisons

une trace pour montrer ce qui a été examiné. Si l'endroit a déja été examiné INCONNU n'a aucun effet. Si chacune des quatre fonctions conduit à une position inconnue, on effectue le RETurn sans essayer d'autre possibilité. Ceci signifie que GAUCHE a priorité sur DROITE, qui a priorité sur HAUT, qui a priorité sur BAS.

Remarquez que cela implique que la souris prendra toujours les mêmes décisions, et que XD% et YD% sont seulement mis à jour si la variable de couleur (CC) a la bonne valeur ce qui indique un chemin possible. (Organigramme 10.2).



Organigramme 10.2: Les coins.

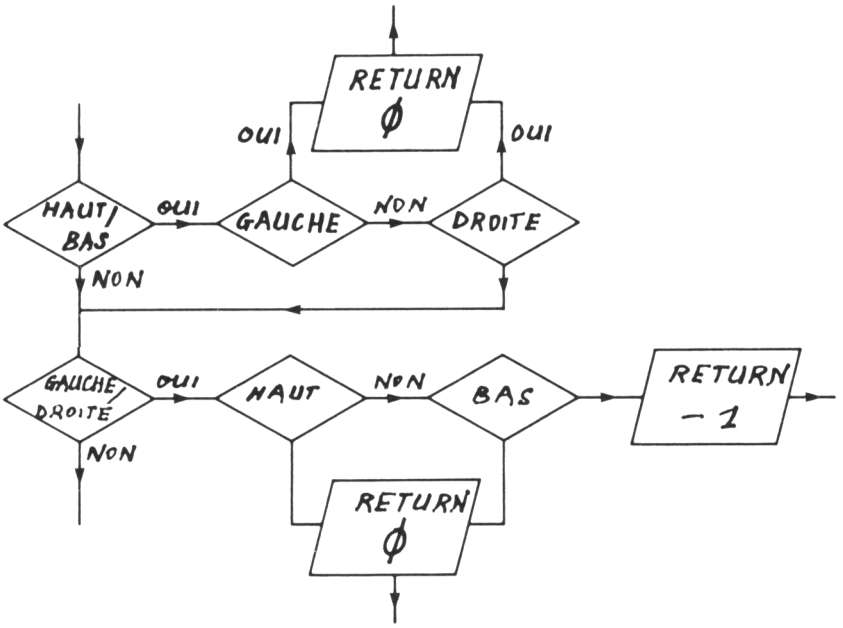
```
150  FICTIF=VERIF
3000 DEFine FuNction VERIF
3020    CC=6
3060      IF GAUCHE THEN RETurn 0
3070      IF DROITE THEN RETurn 0
3160      IF HAUT THEN RETurn 0
3170      IF BAS THEN RETurn 0
3240  RETurn -1
3250 END DEFine VERIF
4000 DEFine PROCedure INCONNU (X1,Y1)
4010   IF MEMORY(X1,Y1)=0 THEN
4020     MEMORY(X1,Y1)=MAZE(X1,Y1)
4030     TRACE X1,Y1,MEMORY(X1,Y1)
4040 END IF
```

```
4050 END DEFine INCONNU

5000 DEFine FuNction GAUCHE
5010   INCONNU X%-1,Y%
5020    IF MEMORY (X%-1,Y%)=CC THEN XD%=-1:YD
%=0:RETurn -1
5030   RETurn 0
5040 END DEFine GAUCHE

6000 DEFine FuNction DROITE
6010   INCONNU X%+1,Y%
6020    IF MEMORY (X%+1,Y%)=CC THEN XD%=1:YD%
=0:RETurn -1
6030   RETurn 0
6040 END DEFine DROITE

7000 DEFine FuNction HAUT
7010   INCONNU X%,Y%-1
7020    IF MEMORY (X%,Y%-1)=CC THEN XD%=0:YD%
=-1:RETurn -1
7030   RETurn 0
7040 END DEFine HAUT

8000 DEFine FuNction BAS
8010   INCONNU X%,Y%+1
8020    IF MEMORY (X%,Y%+1)=CC THEN XD%=0:YD
%=1:RETurn -1
8030   RETurn 0
8040 END DEFine BAS
```

Si vous faites l'essai avec un tracé sinueux comme celui de la Figure 10.1 vous verrez que GAUCHE et DROITE sont testées la plupart du temps, le RETurn s'effectue en général avant d'atteindre HAUT et BAS.

MOINS DE TESTS

Actuellement GAUCHE, DROITE et HAUT sont essayés avant BAS, il serait plus efficace de réduire les tests et d'introduire un peu plus de logique. Quand XD% vaut 0, c'est-à-dire quand la souris se déplace vers le bas ou le haut, il suffit d'essayer GAUCHE ou DROITE. De même quand YD% vaut 0, on essayera HAUT et BAS. (Organigramme 10.3).



Organigramme 10.3: Moins de tests.

```
3030      IF XD%=0 THEN
3060            IF GAUCHE THEN RETurn 0
3070            IF DROITE THEN RETurn 0
3120      END IF
3130      IF YD%=0 THEN
3160            IF HAUT THEN RETurn 0
3170            IF BAS THEN RETurn 0
3220      END IF
```

LES BIFURCATIONS

Notre souris se déplacera correctement dans les coins, mais quand elle atteindra une bifurcation, elle tournera toujours à gauche ou vers le haut si cela est possible, car ces directions sont prioritaires. Ce comportement rigide la conduira à tourner en rond, il serait préférable d'introduire un choix aléatoire entre deux directions possibles, de façon à ce que les priorités ne soient pas toujours les mêmes. Des variables LR% et UD% prennent de façon aléatoire les valeurs 1 et 2, et sont utilisées pour changer l'ordre des priorités. (Organigramme 10.4).

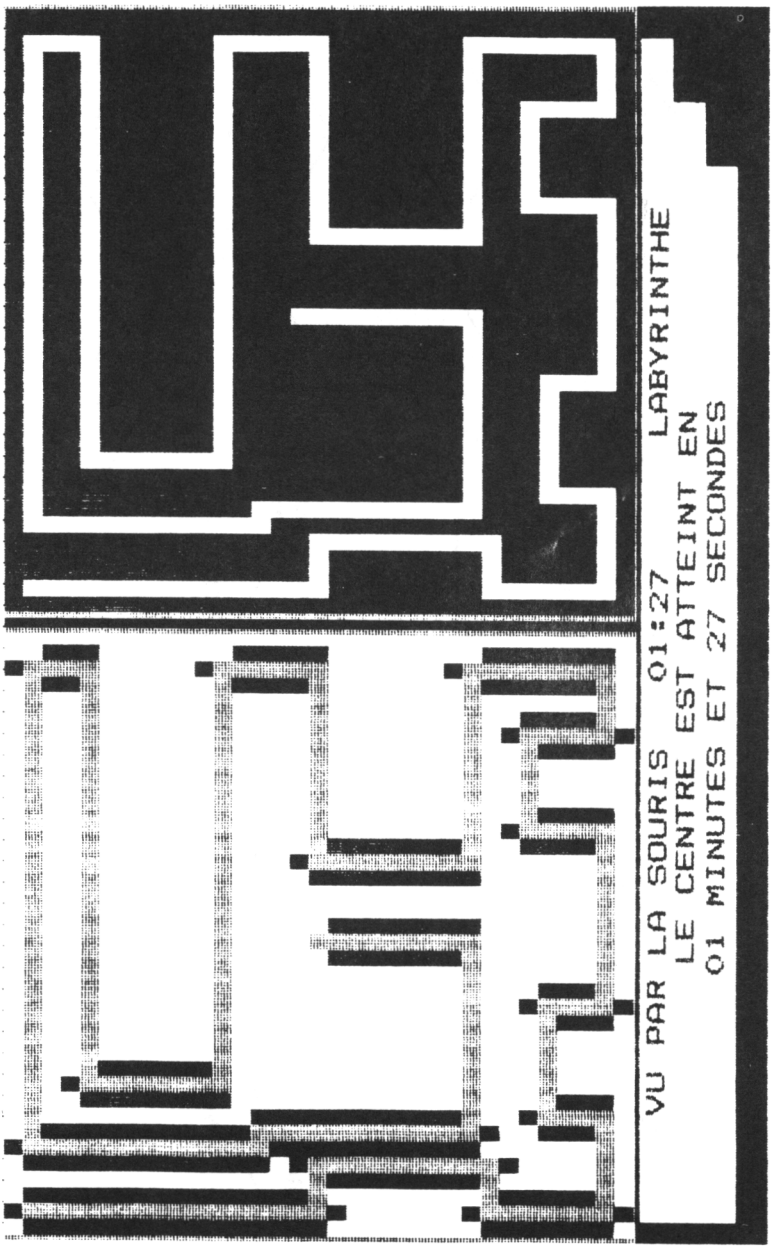
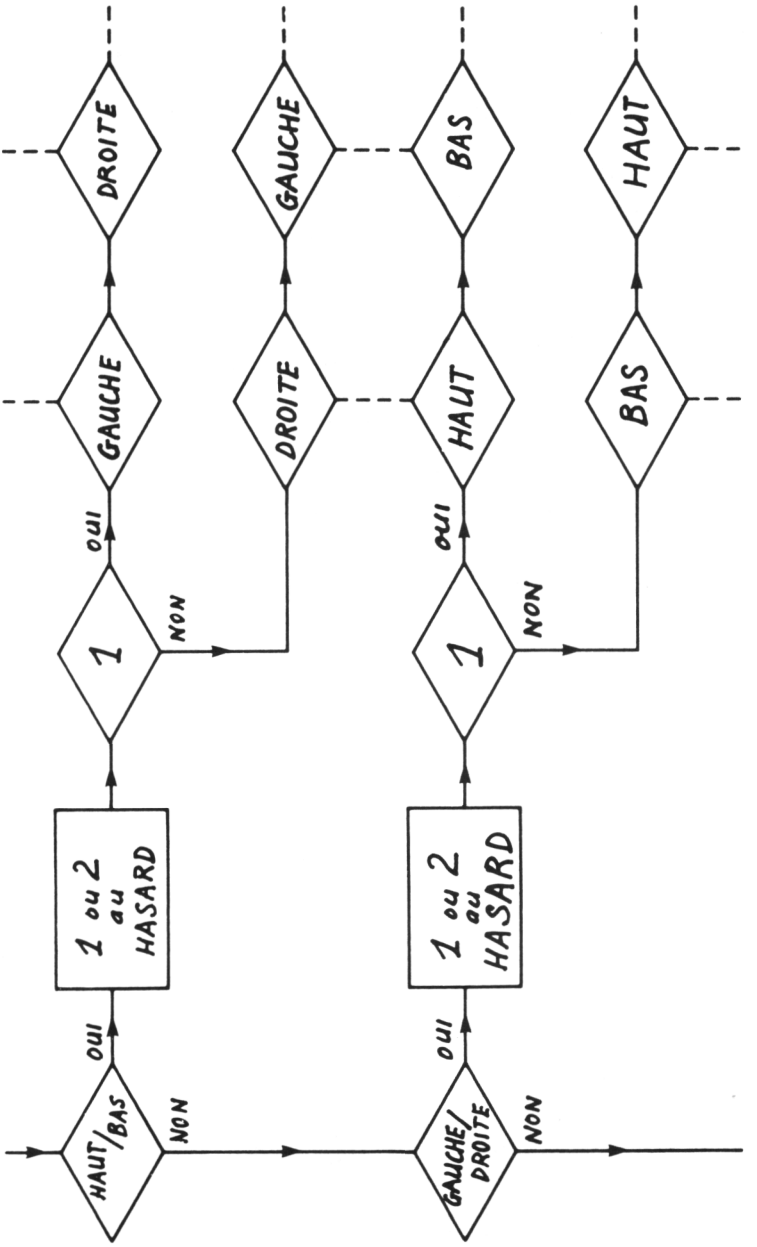


Figure 10.1: Les Coins.



Organigramme 10.4: Les bifurcations.

```
3000 DEFine FuNction VERIF
3020   CC=6
3030     IF XD%=0 THEN
3040          LR%=RND(1 TO 2)
3050             IF LR%=1 THEN
3060                IF GAUCHE THEN RETurn 0
3070                IF DROITE THEN RETurn 0
3080             ELSE
3090                IF DROITE THEN RETurn 0
3100                IF BAS THEN RETurn 0
3110             END IF
3120     END IF
3130     IF YD%=0 THEN
3140       UD%=RND(1 TO 2)
3150         IF UD%=1 THEN
3160                IF HAUT THEN RETurn 0
3170                IF BAS THEN RETurn 0
3180              ELSE
3190                IF HAUT THEN RETurn 0
3210              END IF
3220     END IF
3240   RETurn -1
3250 END DEFine VERIF
```

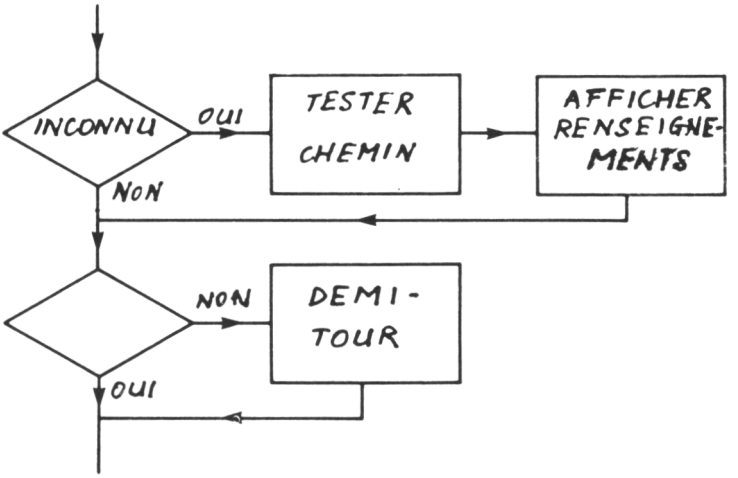
Essayez sur un labyrinthe contenant un carré (Figure 10.2), vous remarquerez que la souris ne suit plus toujours le même chemin.

LES RETOURS EN ARRIERE

Lorsque tous les essais sont négatifs, et lorsque la position suivante du labyrinthe vaut 0 (indication d'un mur), la seule possibilité pour la souris est de revenir sur ses pas. Pour cela on multipliera XD% et YD% par -1. (Organigramme 10.5).

```
150 INCONNU X%+XD%,Y%+YD%
160   IF VERIF AND MAZE (X%+XD%,Y%+YD%)=0 THE
N XD%=XD%*-1:YD%=YD%*-1
```

Elle fera maintenant demi-tour en atteignant un cul-de-sac, et continuera ses recherches jusqu'à ce qu'elle ait trouvé un cheminement possible.



Organigramme 10.5: Retour en arrière.

OU NE VONT PAS LES SOURIS

Il semble raisonnable de donner une plus grande priorité aux parties du labyrinthe qui n'ont pas encore été explorées.

Jusqu'ici les endroits inexplorés sont marqués en blanc (couleur 6), et les endroits connus sont marqués en vert dans MEMORY. On peut s'arranger pour que, lorsque la souris revient sur ses pas, la trace soit rouge au lieu de vert. Ceci se fait facilement en ajoutant une boucle qui change la valeur de CC dans la routine VERIF. Comme le pas de la boucle est de 2, le blanc est testé d'abord, puis le vert, puis le rouge en dernier ressort. Dès que l'égalité est trouvée, on exécute le RETurn; on a donc programmé la priorité suivante: blanc > vert > rouge.

```
3020   FOR C=6 TO 2 STEP -2
3030   END FOR C
```

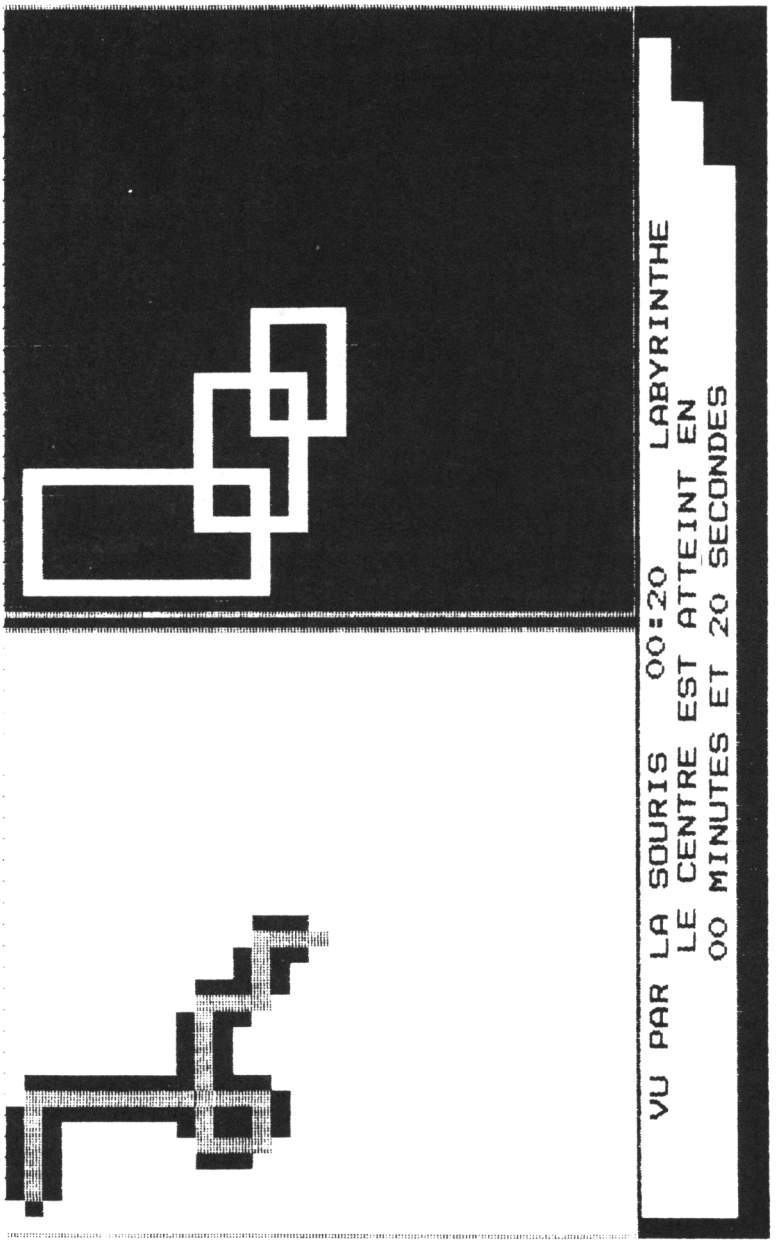


Figure 10.2: Les bifurcations.

LA VITESSE DE CROISIERE

Maintenant les progrès de la souris sont lents mais sûrs, il semble préférable de ne pas faire des investigations trop détaillées, si la voie semble libre devant, (couleur 6).

En ajoutant la routine ci-dessous, la vitesse de la souris augmente énormément, car elle ne fait des tests que si cela est indispensable. (Figure 10.3).

Les inconvénients de cette approche sont que les déplacements sont plus prévisibles, et qu'elle ira tout droit à une bifurcation lorsque c'est possible. En particulier certains cheminements ne seront pas trouvés sans revenir en arrière. Est-ce important? Cela dépend de la situation et de la complexité du labyrinthe, et donc de vous!

```
3010   IF MEMORY (X%+XD%,Y%+YD%)=6 THEN RETur
n 0
```

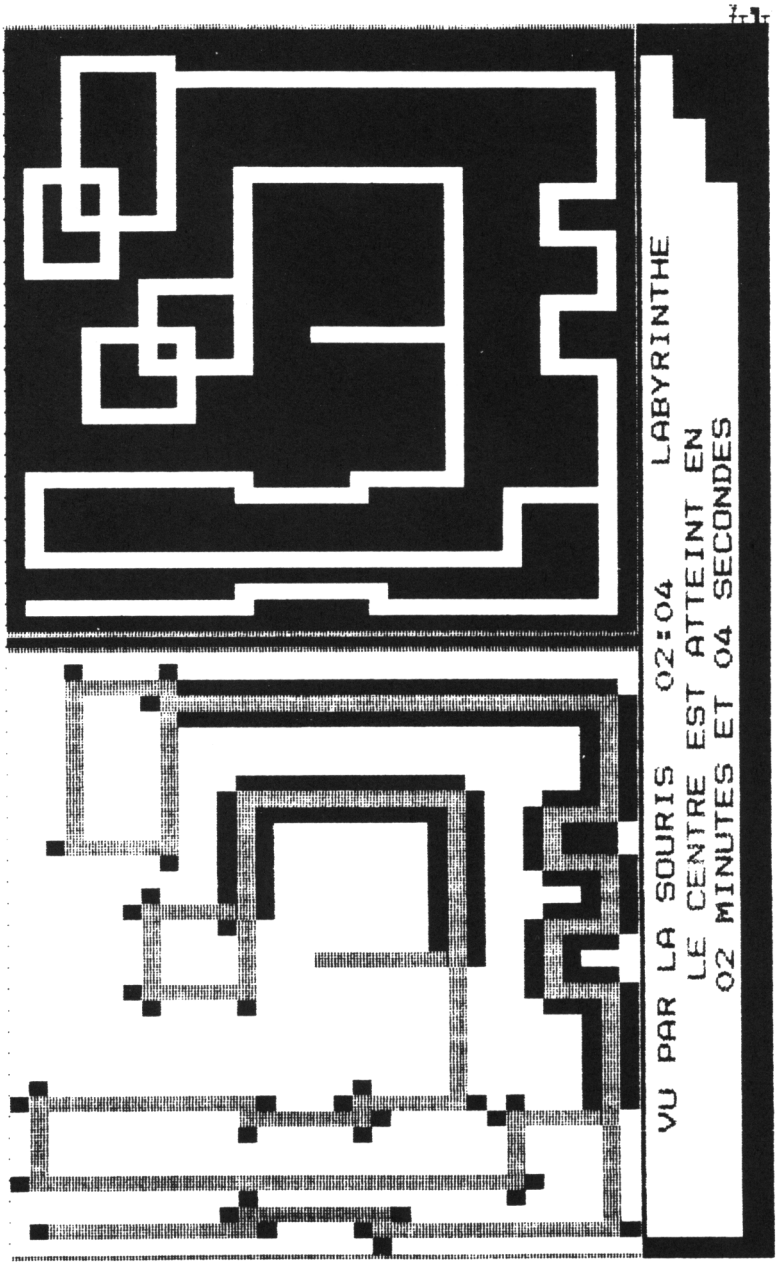


Figure 10.3: La vitesse de croisière.

11

Utilisation intelligente du programme « archive »

Le logiciel ARCHIVE, base de donnée fournie avec le QL est un outil puissant de manipulation de fichiers et d'obtention d'informations. C'est un bon exemple d'un programme intelligent de haute technicité, mais pour l'exploiter avec toutes ses possibilités, il est bon de savoir très clairement comment il fonctionne.

De nombreux livres seront sans doute écrits sur ARCHIVE, mais ici, nous nous concentrerons sur deux aspects particuliers. Le premier est de savoir comment extraire les informations désirées avec précision et le second d'établir un mode d'emploi facile pour le premier venu.

Commençons par charger ARCHIVE avec la commande:

```
lrun mdv1_boot
```

Une fois ceci fait, la commande CAT (CATalogue) d'ARCHIVE sur le microdrive 1 va permettre de visualiser le fichier inclus sur la cassette et destiné à servir d'exemple (PAYS_DBF).

```
cat "mdv1_"
```

(Notez que les guillements sont indispensables pour accéder au microdrive à partir d'ARCHIVE.)

Pour faciliter les choses, nous allons donc utiliser ce fichier (PAYS_DBF), afin d'explorer

le potentiel d'ARCHIVE. Nous avons donc besoin de le consulter:

```
lis "mdv1_pays"
```

Une fois ceci fait, AFFICHE, permettra de visualiser le premier enregistrement du fichier (voir la Figure 11.1).

```
nom logique    : maître
pays$          : AFGHANISTAN
continent$     : ASIE
capitale$      : KABOUL
langues$       : PACHTO,DARI
monnaie$       : AFGHANI
pop            : 19.5
pib            : 110
superficie     : 652
```

Figure 11.1: Premier enregistrement.

```
affiche
```

pour visualiser les autres enregistrements vous pouvez utiliser:

```
affiche
suiv
```

et si vous voulez continuer à voir les enregistrements suivants, vous pouvez éviter de taper 'suiv' à chaque fois en appuyant sur 'F5', qui répète la dernière ligne qui a été entrée au clavier, pui ENTER. Bien que cette facilité soit d'un intérêt limité dans ce cas particulier, elle peut être très utile lorsqu'une longue suite de commandes est nécessaire.

Pour revenir sur vos pas, tapez:

```
préc
```

ou pour des déplacements plus importants:

```
début
```

ou

```
fin
```

RECHERCHE D'UNE CORRESPONDANCE

La commande la plus simple est TROUVE, qui recherche dans toutes les zones de l'enregistrement la chaîne correspondante à celle recherchée.

Exemple:

```
trouve "europe"
```

affichera l'enregistrement ALBANIE, qui est le premier pays d'EUROPE rencontré dans le fichier.

Il est très important de noter que 'europe' a correspondu à 'EUROPE' dans l'enregistrement; TROUVE ne fait en effet pas la différence entre les majuscules et les minuscules.

Pour trouver le prochain enregistrement qui correspond à la chaîne recherchée (europe), il faut taper CONTINUE et non pas SUIV.

Ainsi:

```
continue
```

affichera l'enregistrement correspondant à l'AUTRICHE.

Une des caractéristiques de TROUVE est que cette commande provoque une recherche sur n'importe quelle zone des enregistrements, indépendamment des caractères voisins. Cela peut parfois poser des problèmes:

```
trouve "asie"
```

affiche d'abord AFGHANISTAN puis si l'on continue plusieurs fois (39), PAPOUASIE (Océanie) à cause de "ASIE" contenu dans PapouASIE.

D'un côté, cela peut aussi être un avantage, si vous désirez trouver seulement une partie d'un enregistrement. La zone 'langues$', par

exemple, contient parfois plusieurs langues et si vous désirez n'en trouver qu'une seule TROUVE fonctionnera parfaitement:

ainsi:

```
trouve "ANGLAIS"
```

vous donnera AFRIQUE DU SUD où les trois langues sont "BANTOU, AFRICAANS, ANGLAIS", et non pas simplement 'ANGLAIS'.

On pourrait aussi chercher les pays qui utilisent le Dollar ($) comme monnaie:

```
TROUVE "$"
```

s'arrête d'abord sur AUSTRALIE (avec le $ australien).

Il peut parfois présenter un avantage de tronquer la chaîne recherchée pour obtenir toutes les correspondances désirées. Si vous comparez:

```
trouve "anglais"
```

avec

```
trouve "angl"
```

vous verrez que la deuxième solution augmente d'une façon significative les réponses.

RECHERCHES SPECIFIQUES

CHERCHE est une commande plus spécifique et plus performante qui demande qu'une condition particulière soit vérifiée. Elle n'agit que sur les zones spécifiées et fait la différence entre les majuscules et les minuscules. Ainsi:

```
cherche continent$="asie"
```

ne s'arrête sur aucun enregistrement alors que:

```
cherche continent$="ASIE"
```

s'arrête, en revanche:

```
cherche continent$="AMERIQUE"
```

ne fonctionnera pas non plus du fait que 'AMERIQUE' est toujours suivie de quelque chose, comme par exemple "DU NORD".

Bien que vous ne puissiez pas chercher les pays qui utilisent le Dollar comme monnaie (puisque le signe $ est toujours suivi de quelque chose), vous pouvez aisément déterminer si un nombre est plus grand ou plus petit qu'un autre. Ainsi,bien que CHERCHE soit en théorie plus exact , il ne vous permet pas d'être aussi précis!

Ainsi:

```
cherche superficiel<2
```

donne

```
HONG KONG (1)
```

et

```
cherche superficiel >10000
```

ne donne que

```
U.R.S.S. (22402)
```

Les chaînes peuvent être comparées comme les nombres:

```
cherche pays$>"C"
```

donne

```
CAMEROUN
```

(le premier pays commençant par une suite de caractères supérieurs - dans l'ordre alphabétique - à la seule lettre spécifiée 'C'.)

Plusieurs conditions peuvent être combinées; par exemple, quels sont les pays d'Afrique dont le Français est la langue?

```
cherche continent$="Afrique" et
langues$="Français"
```

BENIN
COMORES
CONGO
COTE D'IVOIRE
GUINEE
MALI
REP. CENTRAFRICAINE
REUNION
TOGO

Et pourquoi ne pas demander quels sont ceux qui utilisent le Franc CFA parmi ceux-là? La solution la plus évidente consiste à rajouter une condition supplémentaire à remplir:

```
cherche continent$= "AFRIQUE" et langues$=
"FRANCAIS" et monnaie$= "FRANC CFA"
```

BENIN
COMORES
CONGO
COTE D'IVOIRE
REP. CENTRAFRICAINE
TOGO

ISOLEMENT DE CERTAINS ENREGISTREMENTS

Un moyen plus efficace de traiter ce genre de problèmes est d'isoler un certain nombre d'enregistrement momentanément. Le nombre total d'enregistrements du fichier PAYS peut être visualisé ainsi:

```
écris nombenr()
```

qui donne 150.

Vous pouvez isoler les pays d'Afrique de la

façon suivante:

```
isole continent$="AFRIQUE"
```

maintenant

```
écris nombenr()
```

qui donne seulement 49, du fait que ARCHIVE se comporte comme s'il n'y avait plus que ces enregistrements qui existaient désormais. Maintenant:

```
cherche langues$="FRANCAIS"
```

cherchera les pays dont la langue est le Français et seulement en Afrique. Vous pourriez aussi procéder ainsi:

```
isole langues$="FRANCAIS"
écris nombenr()
```

qui ne laisse plus que 9 pays dans le fichier. (Cette sélection peut aussi se faire en une seule fois.)

METTRE LES CHOSES EN ORDRE

Si vous regardez la liste des pays d'Afrique dont la langue est le Français ci-dessus, vous constaterez qu'ils sont en ordre alphabétique. Cela est entièrement fortuit et vient seulement du fait que le fichier original a été déjà ordonné. Cet ordre peut être cependant facilement modifié.

Ainsi:

```
trie superficie¡a
```

les trie en ordre croissant en fonction de leur superficie:

COMORES
REUNION

TOGO
BENIN
GUINEE
COTE D'IVOIRE
CONGO
REP. CENTRAFRICAINE
MALI

et

```
trie capitale$;a
```

les trie en ordre croissant selon leur capitale (avec ABIDJAN en COTE D'IVOIRE en tête de liste).

Lequel de ISOLE ou CHERCHE est le plus rapide dépend vraiment de ce que vous recherchez.

Pour restituer le fichier en entier tapez:

```
restaure
```

CORRESPONDANCES PARTIELLES

Dans un des exemples précédents nous avons isolé les pays africains dont le Français est la langue - mais qu'en est-il de ceux qui ont le Français et une autre langue? Souvenez-vous que TROUVE n'est pas spécifique - pourquoi ne pas isoler les pays d'Afrique, comme précédemment, et TROUVEr 'FRANCAIS'.

```
isole continent$="AFRIQUE"
trouve "FRANCAIS"
continue
```

UTILISATION DE PROCEDURES

Jusqu'ici nous avons fait qu'effleurer le potentiel d'ARCHIVE en nous limitant aux commandes directes, qui n'ont fait que retrouver et afficher les enregistrements correspondants.

En utilisant l'éditeur nous pouvons adapter des PROCedures pour des tâches plus spécifiques. Pour entrer dans le mode éditeur tapez:

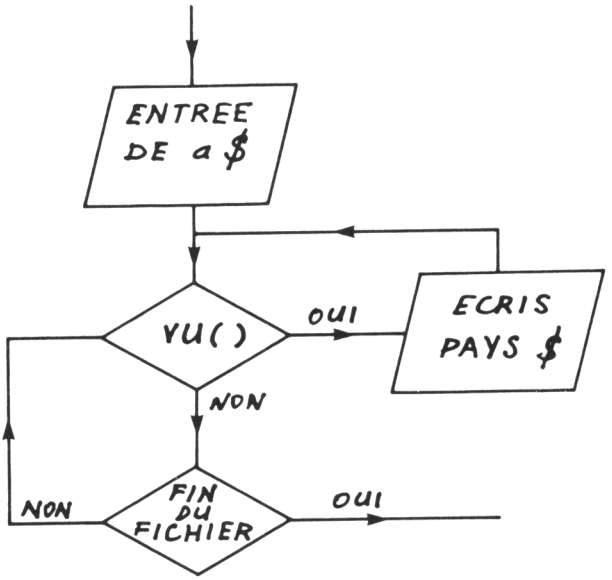
```
édite
```

et lorsque vous êtes invités à entrer un nom de procédure, tapez:

```
cont1
```

puis les lignes suivantes qui effectuent un TROUVE automatique.

```
proc cont1
```



Organigramme 11.1: PROC cont 1.

La chaîne a$ est initialisée lors de la réponse à la question 'quel continent?', et la recherche (TROUVE) s'effectue tant que la condition vu<> est vérifiée (voir l'organigramme 11.1). En fait vu<> vaut zéro lorsque le dernier TROUVE n'aboutit pas - il n'y a plus de correspondance entre la chaîne recherchée et les enregistrements. Notez que nous avons spécifié que seul les noms des pays devaient s'afficher à l'écran et non pas l'ensemble de l'enregistre-

ment, de sorte que seule l'information demandée apparaît. Une fois que la PROCedure a été rentrée, vous pouvez taper ESC pour revenir à ARCHIVE et exécuter votre PROCedure en tapant simplement son nom:

```
cont1
```

Lorsque vous êtes invités à rentrer un nom tapez 'asie' et aussitôt apparaît la liste des noms de pays en Asie - mais remarquez que l'URSS (dont une partie est en Asie) a aussi été sélectionnée.

Pour restreindre la correspondance à 'ASIE' vous pouvez CHERCHEr. La seule ligne qui doit être changée est:

```
trouve a$
```

qui devient

```
cherche continent$=a$
```

Bien entendu, seules les majuscules provoqueront une correspondance, ce qui peut parfois être gênant. Pour contourner ceci, il suffit de convertir systématiquement toute entrée au clavier en majuscule, et une procédure que nous appelerons CONVERT. Appelez l'éditeur comme auparavant et tapez sur 'F3' et C pour créer une nouvelle PROCedure.

```
proc convert
```

La ligne d'entrée de CONT1 doit maintenant être remplacée par :

```
écris "Quel contient?";:convert
```

pour revenir à la PROCedure CONT1, tapez sur ESC puis sur SHIFT et TABULATE, puis modifiez la ligne.

UNE (INTER) FACE PLUS PLAISANTE

Jusqu'ici vous avez tapé la chaîne à rechercher exactement comme elle apparaît dans l'enregistrement - mais ça serait plus agréable si vous pouviez être plus vague. Qui sait, vous en arriveriez peut-être à trouver que les ordinateurs valent la peine d'être pris en considération!

Nous allons définir une nouvelle PROCedure appelée DIT, qui fournira un 'habillage' de sorte que l'utilisateur n'aura pas besoin de se soucier de la cuisine interne d'ARCHIVE (voir organigramme 11.2).

Une des caractéristiques principales est qu'elle utilise une recherche à l'aide de la fonction souschn () (sous-chaîne), pour trouver ce que vous cherchez, au lieu de l'accepter comme il est entré.

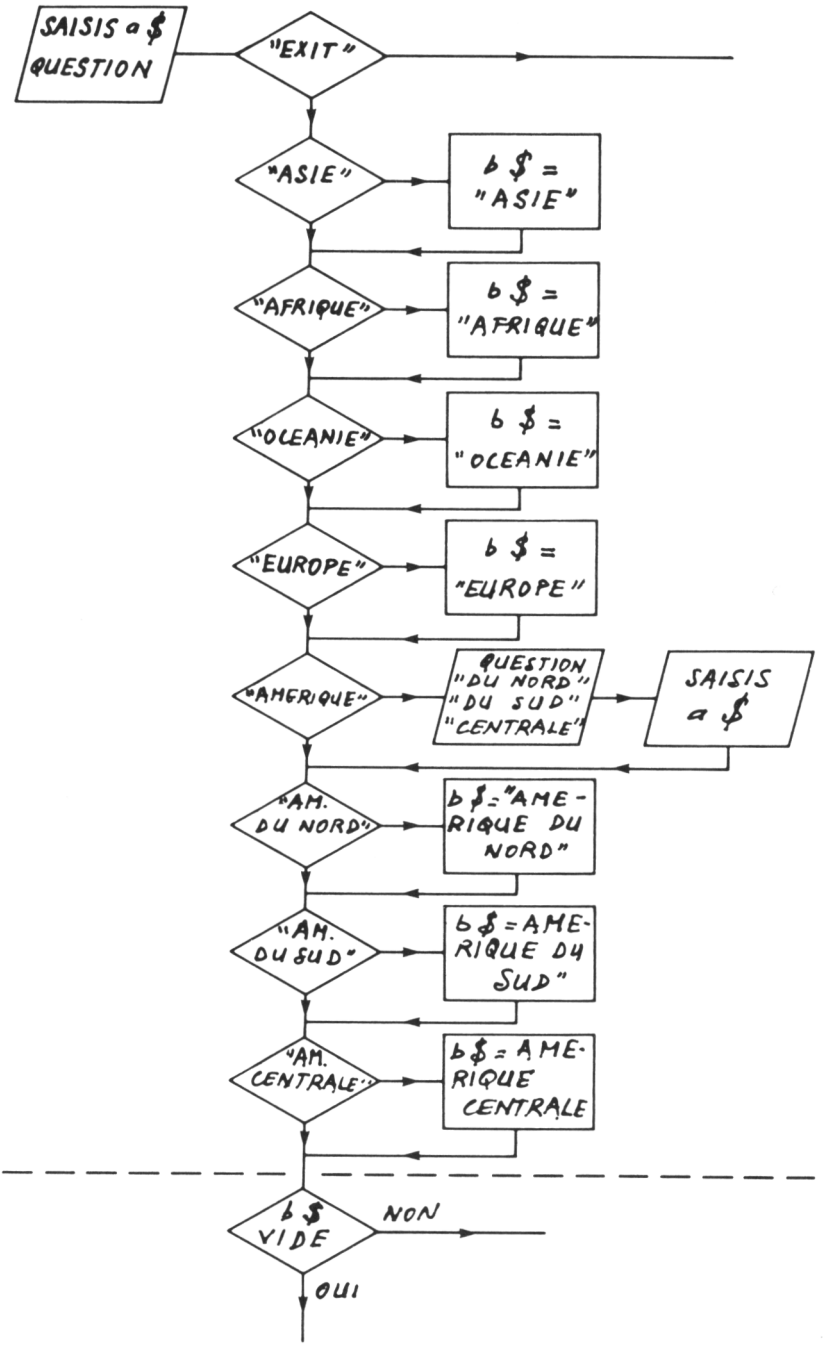
Maintenant, lorsque vous utilisez DIT, vous êtes invités à entrer une chaîne qui peut être ce que vous voulez. La procédure vérifie si les mots comme ASIE, AFRIQUE, OCEANIE, et EUROPE sont présents. Si vous entrez n'importe quelle phrase contenant un de ces mots-clés décrivant des continents, une recherche est faite pour retrouver une correspondance adéquate.

Ainsi:

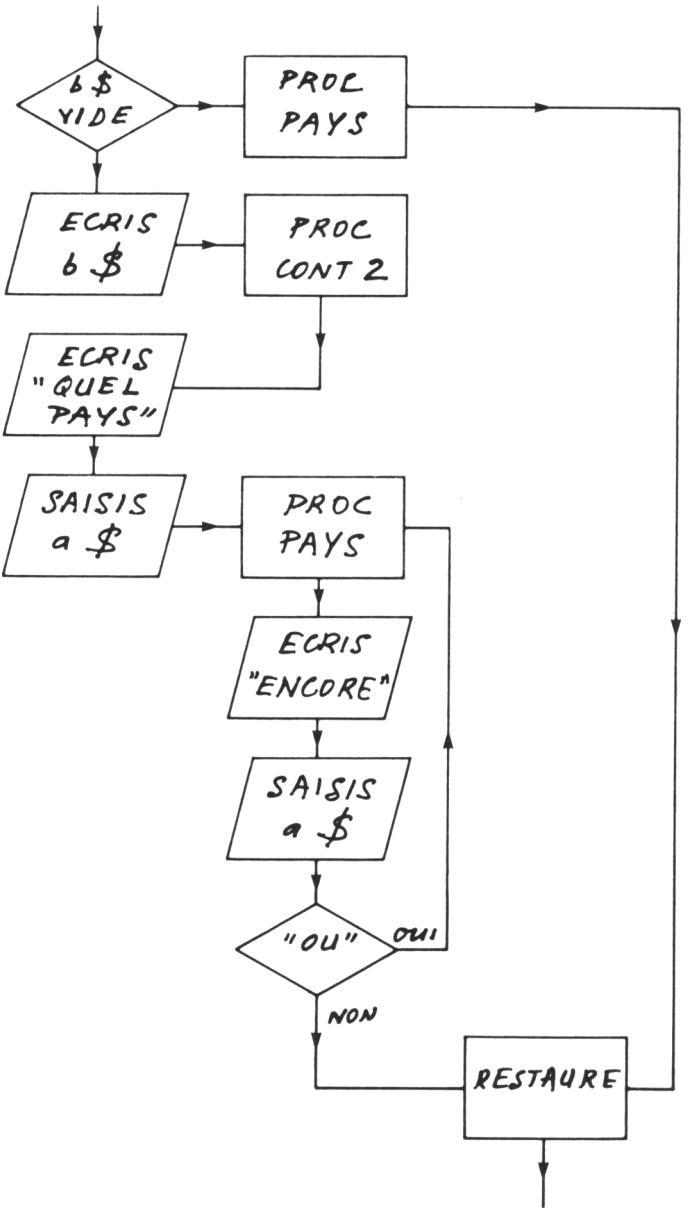
ASIE
ASIATIQUE
AFRIQUE
AFRICAIN
OCEANIE
OCEANE
EUROPE
EUROPEEN

seront tous acceptés

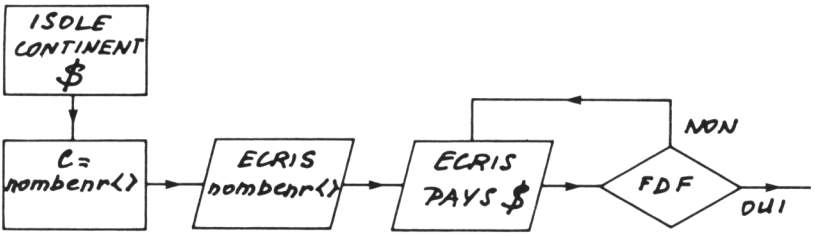
Dans le cas de l'AMERIQUE une approche plus fine est utilisée étant donné que le fichier



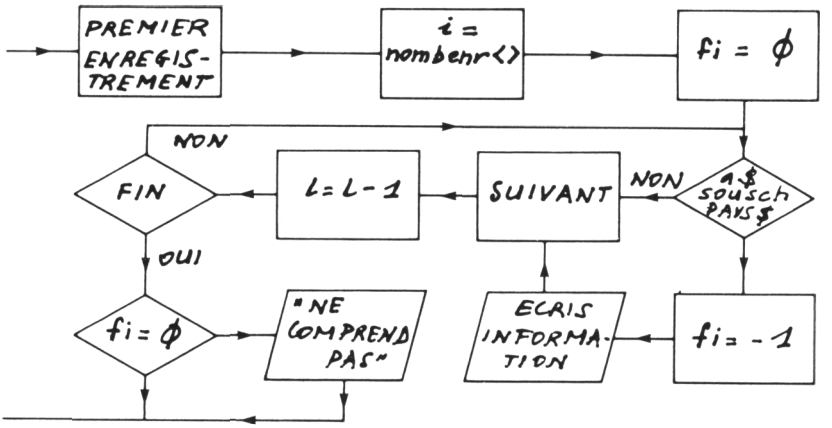
Organigramme 11.2(a): PROC DIT.



Organigramme 11.2(b): PROC DIT (ii).



Organigramme 11.2(c): PROC cont 2.



Organigramme 11.2(d): PROC Pays.

PAYS la divise en trois secteurs qui doivent être spécifiés précisément. Notez que cela dépend de l'espace après AMERIQUE.

Si le nom d'un continent n'est pas trouvé (b$=""), la PROCedure PAYS vérifie qu'il ne s'agit pas d'un nom de pays en contrôlant votre entrée avec tous les pays du fichier.

Si une correspondance est détectée, un certain nombre de variables sont extraites de l'enregistrement et présentées clairement dans du texte. Notez qu'apparaît aussi la densité de population dérivée des variables proposées, alors que cette information n'existe pas dans les enregistrements originaux. La boucle TANTQUE L (c'est-à-dire nombenr()) lit tous les enregistrements du fichier, de sorte que vous pouvez demander des informations sur plus d'un pays à la fois, mais la fonction clavier vous donne un bon moyen d'arrêter si vous ne pouvez

plus supporter le bruit des microdrives plus longtemps! Si aucune correspondance n'est détectée, un message apparaît et vous retournez à la procédure d'appel.

Lorsque le nom d'un continent est détecté, la procédure CONT2 est appelée et sélectionne tous les pays de ce continent et affiche leur nom. On vous demande alors "sur quel pays vous désirez des informations", et la PROCedure PAYS est alors utilisée pour trouver ceci comme auparavant. La boucle TANTQUE R vous permet de faire d'autres recherches sur ce continent.

L'Organigramme 11.2 donne un exemple de ce que vous pouvez voir sur votre écran.

Cette approche est incontestablement plus plaisante, et vous pouvez voir que le travail le plus dur a dû être fait en avance et que la recherche avec une sous-chaîne ralentit inévitablement les choses. Vous pouvez maintenant peut-être améliorer ce programme afin de pouvoir consulter plusieurs zones en même temps.

```
proc CONT1
éponge
écris "QUEL CONTINENT? ";:convert
trouve a$
tantque vu()
écris pays$;" ";
continue
ftantque
fproc

proc cont2
isole continent$=b$
que c=nombenr()
écris
écris "Il y a ";c;" pays en ";b$
écris
tantque c
si C/4=ent(C/4)
écris
fsi
écris pays$; "     ";
suiv
```

```
que c=c-1
ftantque
fproc


proc convert
saisis a$
que a$=majus(a$)
fproc
proc dit
restaure
mode 0,6
que fictif=-1
tantque fictif
éponge
écris
écris "INFORMATIONS MONDIALES"
écris
écris "ENTREZ VOTRE QUESTION SVP"
écris
que B$=""
écris
écris "? ";
convert
si A$="STOP"
écris "A BIENTOT"
que fictif$=clavier()
mode 1,6
restaure
reviens
fsi
si souchn(A$,"ASI")>0
que B$="ASIE"
fsi
si souchn(A$,"AFRI")>0
que b$="AFRIQUE"
fsi
si souschn(A$,"OCEAN")>0
que B$="OCEANIE"
fsi
si souschn(A$,"EUROPE")>0
que B$="EUROPE"
fsi
si souschn(A$,"AMERIQUE")>0
écris
écris "PRECISEZ"
écris "AMERIQUE DU NORD"
```

```
écris "AMERIQUE DU SUD"
écris "OU AMERIQUE CENTRALE"
écris
convert
fsi
si souschn(A$,"AMERIQUE DU NORD")>0
que B$="AMERIQUE DU NORD"
fsi
si souschn(A$,"AMERIQUE DU SUD")>0
que B$="AMERIQUE DU SUD"
fsi
si souschn(A$,"AMERIQUE CENTRALE")>0
que B$="AMERIQUE CENTRALE"
fsi
si B$()""
écris b$
cont2
que r=-1
tantque r
écris
écris "Sur quel pays désirez-vous d'autres
informations"
convert
écris
pays
écris "Voulez-vous avoir"
écris "d'autres informations sur ";b$
convert
si souschn(a$,"OU")=0
que r=0
fsi
ftantque
sinon
pays
fsi
que fictif$=clavier()
restaure
ftantque
fproc
proc pays
début
queL=nombenr()
que FL=0
tantque L
si souschn(A$,pays$)>0
que FL=-1
écris
```

```
encre 4
écris pays$;" est en ";continent$
écris "a une population de ";pop;" millions,"
écris "s'étend sur ";superficie;"000 km2"
écris "(une densité de ";
écris chn(pop/(superficie/1000),0,2);"/ km2)"
écris "La capitale est ";capitale$;","
écris "et la monnaie est le ";
écris monnaie$
encre 7
fsi
suiv
que L=L-1
si touche ()()""
reviens
fsi
ftantque
si FL=0
encre 2
papier 7
écris "Je ne comprends pas ce que vous voulez
dire"
encre 7
papier0
fsi
fproc
```

12

Un vendeur expert

Dans les chapitres précédents, nous avons traité les différents aspects de l'intelligence artificielle. Dans le présent chapitre, nous les avons regroupés dans un seul et unique programme. Un des premiers programmes "intelligents" fut ELIZA qui proposait une thérapie pseudo-psychiatrique; pour notre part nous avons résisté à la tentation de continuer dans cette direction et avons opté pour la réalisation d'un "vendeur artificiel".

Bien que les programmes du type ELIZA, qui tiennent véritablement une "conversation" avec leurs utilisateurs soient assez répandus, le programme suivant est assez original, en ce sens qu'il combine le traitement de la langue naturelle avec un système expert pour produire un résultat qui devrait à la fois comprendre vos demandes faites en langue naturelle et faire des suggestions qui tiennent compte de vos requêtes, des forces et faiblesses des différentes machines selon 20 critères distincts ainsi qu'un nombre de faits commerciaux comme le prix de revient et la marge.

Assez de mots et de valeurs ont été inclus pour rendre le programme intéressant, mais vous pouvez bien sûr le modifier pour l'ajuster à vos utilisations; il suffit pour cela de modifier les lignes DATA. (Les données incluses dans le programme n'ont qu'une valeur démonstrative et nous ne saurions être tenus pour responsables des jugements totalement arbitraires émis par l'ordinateur.)

Le programme en lui-même suit les mêmes méthodes que celles employées précédemment dans ce livre et les fonctions des différentes PROCedures, FuNctions, variables et tableaux sont données dans le tableau 12.1.

VARIABLES SIMPLES

QP%	Nr. de phrases (pour les questions)
Q%	Nr. de questions
R%	Nr. de règles
OB%	Nr. de compléments d'objet
AJ%	Nr. d'adjectifs
AV%	Nr. d'adverbes
LI%	Nr. d'opinions positives
DL%	Nr. d'opinions négatives
NJ%	Nr. d'adjectifs négatifs
NV%	Nr. d'adverbes négatifs
HM%	Nr. de cher/bon marché
BB%	Balance de la banque
CO%	Nr. d'ordinateurs
FE%	Nr. de caractéristiques
CT%	Nr. de coûts
CS%	Nr. de suggestions de coûts
EX%	Nr. d'excuses
HI%	Nr. de suggestions chères
LO%	Nr. de suggestions bon marché
LD%	Aime/n'aime pas
TC%	Coût total
tp%	Profit total
OF%	"Flag" compléments d'objet
NP%	Négatif/positif
M%	"Marker"
OM%	"Marker" du complément d'objet
S1%	Position de "et"
S2%	Position de "mais"
CM%	Position de la virgule
SP%	Position de la recherche
ST%	Départ de la recherche
PH%	Sélection de la question
IS%	Position de la recherche
RU%	Mise à jour des règles
XX%	Cher/bon marché
TX%	Sélection de l'excuse
PT%	Avertissement sur le budget
TS%	Coût du plus cher
BS%	Coût du moins cher

HI%	Coût du plus cher
LO%	Coût du moins cher
SE%	Plus/moins
SL%	Coût de la phrase de sélection

TABLEAUX

OB%(OB%,10)	Compléments d'objet
AJ$(AJ%,6)	Adjectifs
NJ$(NJ%,7)	Adjectifs négatifs
AV$(AV%,6)	Adverbes
NV$(NV%,6)	Adverbes négatifs
LI$(LI%,7)	Aime
DL$(DL%,7)	N'aime pas
Q$(Q%,20)	Compléments d'objet des questions
QP$(QP%,16)	Phrases des questions
CR(Q%)	Coût
PR(Q%)	Profit
IC(Q%)	Coût total
IP(Q%)	Profit total
HM$(HM%,20)	Cher/bon marché
R(R%)	Règle désirée
CO$(CO%,FE%)	Noms des ordinateurs
FE(CO%,FE%)	Noms des caractéristiques
C(CT%)	Evaluation des coûts
CS$(CS%,100)	Suggestions des coûts
EX$(EX%,100)	Excuses
HI$(HI%,100)	Messages hauts
LO$(LO%,100)	Messages bas

Tableau 12.1(a): Variables et tableaux.

PROCEDURES

ECRAN	Initialise les fenêtres
TITRE	Affiche le titre
INIT	Initialise les DATAs
SAISIS_QUESTION	Sélectionne la question (PH$)
TROUV_et	Recherche les "&" dans PH$
JOIN_1	Forme la question avec le complément d'objet à la fin
JOIN_2	Forme la question avec le complément d'objet au milieu
ET_ou_MAIS	Met les règles à jour selon le mot précédent le ET ou MAIS
OUI_PRESENT	Met les règles à jour si OUI est présent dans votre phrase

NON_PRESENT	Effectue la même tâche pour NON
PAS_PRESENT	Met les règles à jour si PAS est présent
DOUBLE_NEGATION	Recherche les doubles négations
AIME	Recherche les verbes positifs
AIME_PAS	Recherche les verbes négatifs
OBJETS	Recherche les compléments d'objet
ADVERBES	Recherche les adverbes positifs
ADVERBES_ NEGATIFS	Recherche des adverbes négatifs
ADJECTIFS	Recherche des adjectifs positifs
ADJECTIFS_ NEGATIFS	Recherche des adjectifs négatifs
CHER_BONMARCHE	Affiche les messages "CHERS"
MISE_A_JOUR	Remet les règles à jour
COUT_PROFIT	Calcule les coûts et profits
DEPENSE	Compare avec la balance de la banque
SAISIS_COMPUTER	Sélectionne l'ordinateur

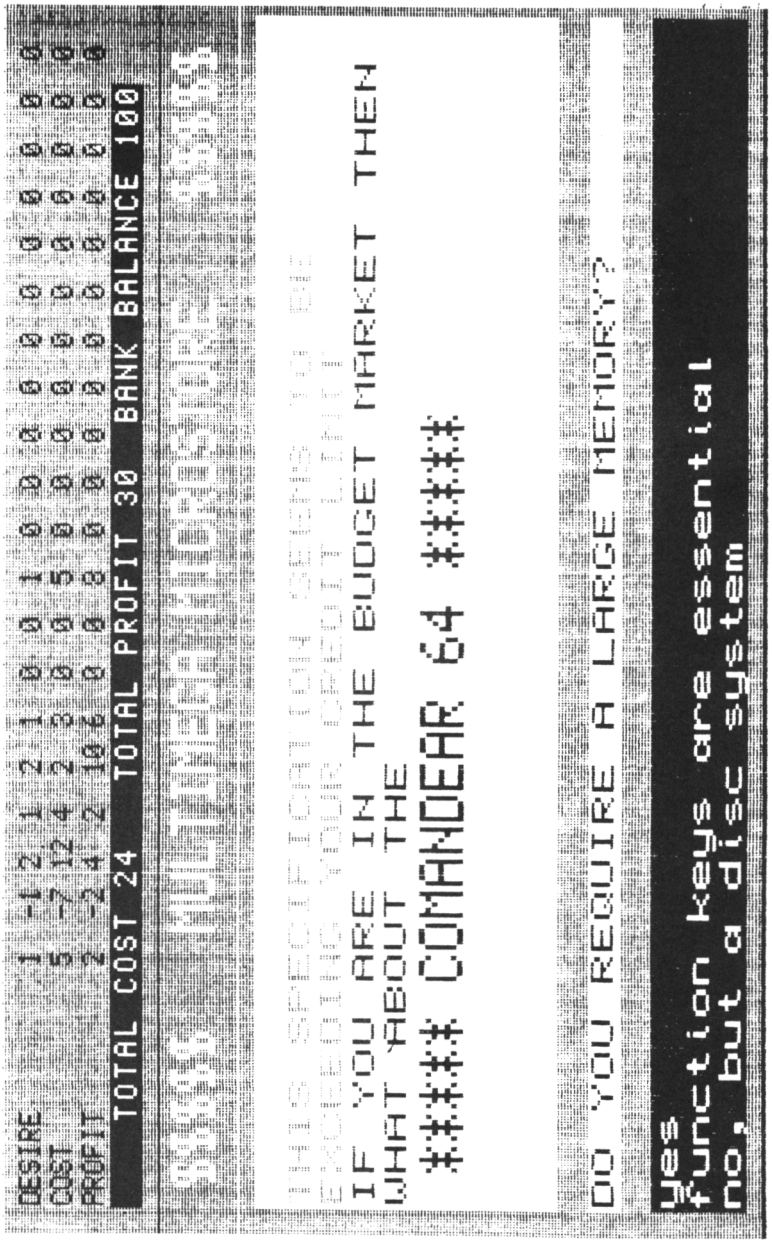
FONCTIONS

TROUVE_slash	Recherche les "/" dans PH$
TROUVE_ast	Recherche les "*" dans PH$
TROUVE_virg	Recherche les "," dans CHAINE$
TROUVE_ET	Recherche les "ET" dans CHAINE$
TROUVE_MAIS	Recherche les MAIS" dans CHAINE$

Tableau 12.1(b): PROCedures et FuNctions.

CONVERSATION

Le programme vous propose de donner votre point de vue sur chacune des caractéristiques que peut posséder n'importe quel micro-ordinateur (les questions exactes sont SAISIeS au hasard dans une liste de QUESTIONS possibles). Notez que les mots-clés ou expressions sont insérés dans la phrase à l'endroit où cela est nécessaire. Ceci est effectué notamment grâce aux PROCedures TROUV_slash, TROUV_et, TROUV_ast, JOIN_1 et JOIN_2.


DESIRE 1 -1 2 1 2 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0
COST 5 -7 12 4 2 3 0 0 5 0 0 0 0 0 0 0 0 0 0 0
PROFIT 2 -2 4 2 10 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
TOTAL COST 24 TOTAL PROFIT 30 BANK BALANCE 100
IF YOU ARE IN THE BUDGET MARKET THEN
WHAT ABOUT THE
**** COMMODEAR 64 ****
DO YOU REQUIRE A LARGE MEMORY?
yes
function keys are essential
no, but a disc system

L'écran est divisé en 5 fenêtres horizontales (Figure 12.1) qui sont affectées à des tâches différentes. La fenêtre #0 (du bas) est destinée à recevoir vos phrases, qui sont entrées en réponse à la question qui apparaît dans la petite fenêtre (#3) juste au-dessus. La fenêtre #4 contient simplement une "publicité" pour "SANA ELECTRONIQUE", alors que la fenêtre du haut (#2) montre une partie du "cerveau du vendeur", afin de montrer selon quelles règles il fonde son jugement. Cela vous donne plus de détails sur la motivation des vendeurs.

Votre phrase est examinée en détails (selon une série de mots-clés) et le tableau R(R%) est mis à jour selon vos requêtes (vous pouvez d'ailleurs le voir en haut de l'écran). Notez que la plupart des mots-clés sont tronqués de sorte que n'importe quelle forme de ces mots peut être reconnue. Si vous êtes obsédé par un critère particulier (ex. un processeur 16 BITS) le vendeur ne vous prend pas tellement au sérieux étant donné qu'il s'agit vraisemblablement d'une "mode" venue d'on ne sait quel "Ordinateurs Magazine", de la semaine précédente!

Le test le plus simple est de vérifier si OUI ou NON sont PRESENTS dans la phrase entrée; ceci a pour effet de soustraire ou d'ajouter 1 au tableau des désirs (R(R%)), et si vous mentionnez le nom de l'OBJET (ex: le GRAPHISME), ceci incrémente encore une fois de un le tableau des désirs. En outre, le fait d'utiliser des adjectifs ou des adverbes négatifs opère la fonction inverse. Le fait de séparer les mots en différents groupes vous permet d'effectuer plus d'un changement à la fois dans le tableau.

Ainsi:

OUI	ajoute 1
OUI DU BASIC	ajoute 2
OUI LE BASIC EST ESSENTIEL	ajoute 3

OUI UN BON BASIC EST ESSENTIEL ajoute 4

Alors que:

NON soustrait 1

et

PAS DE MEMOIRE soustrait 2

En outre, les verbes sont regroupés en AIME et AIME PAS; les derniers inversent l'action du reste des mots.

JE DETESTE LES MACRODRIVES soustrait 1

NON_PRESENT et PAS_PRESENT sont identifiés et la plupart des DOUBLES_NEGATIONS sont correctement interprétées.

Ainsi:

JE N'AIME PAS LE SON soustrait 2

JE NE DETESTE PAS LE SON ajoute 2

Si quelque chose apparaît au début de la phrase et est suivi par une virgule alors TROUVE_virg le supprime et il est ignoré.

Ainsi:

NON, JE NE VEUX PAS D'UN BON SON soustrait 3

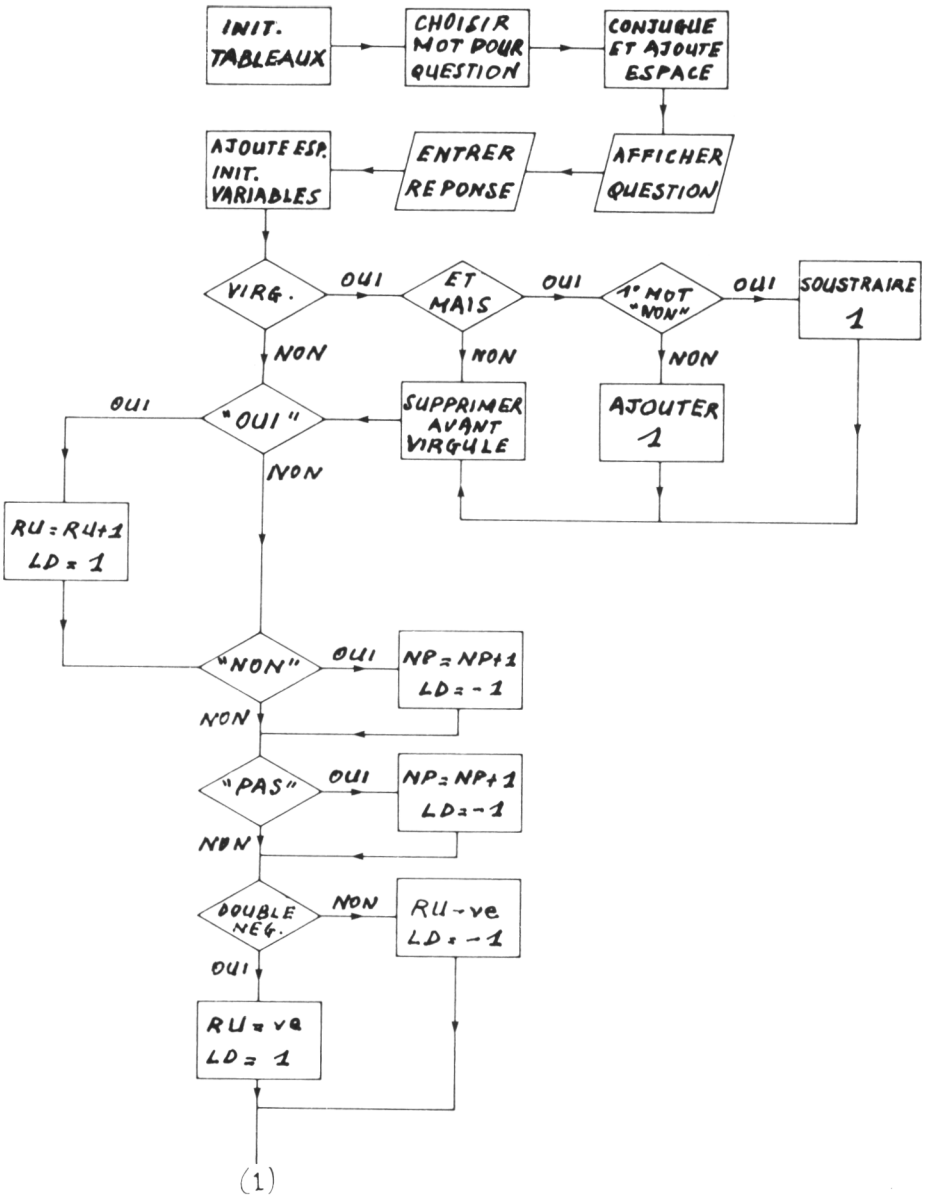
La seule exception, c'est lorsque ET_ou_MAIS sont présents dans la phrase; dans ce cas, les deux parties de la phrase sont traitées indépendamment.

Ainsi, si la question est:

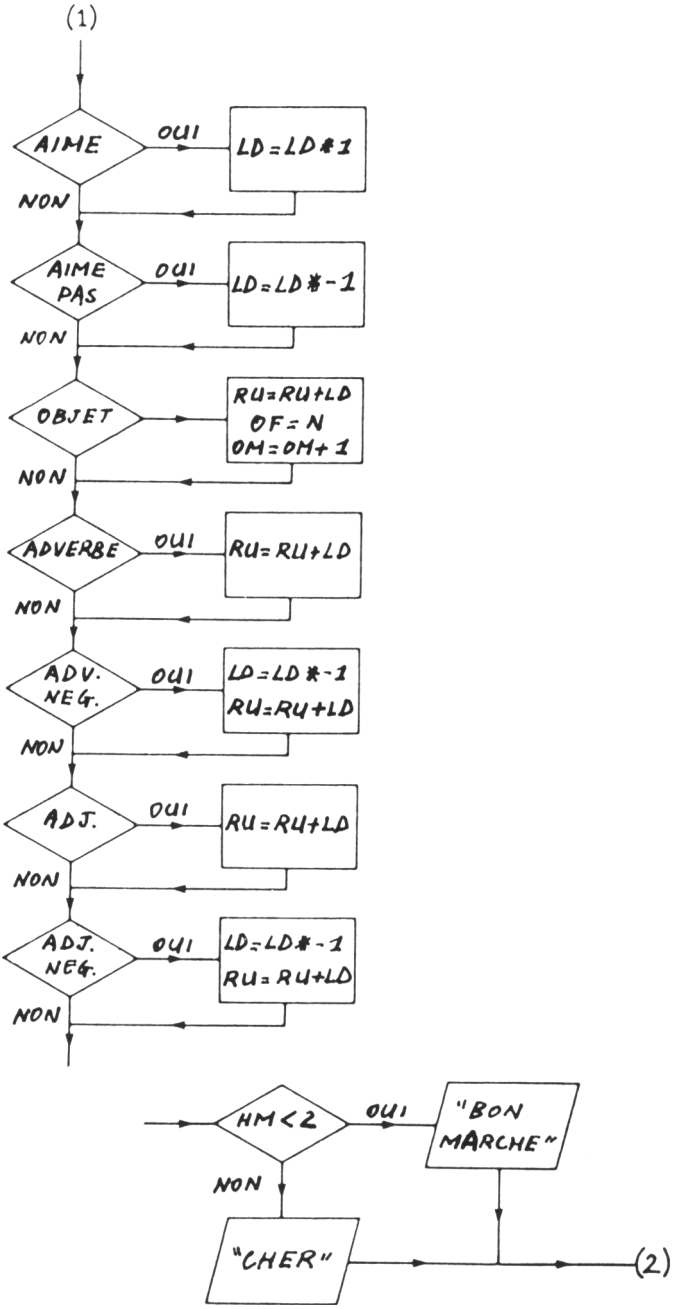
VOULEZ-VOUS DU GRAPHISME?

et que la réponse est:

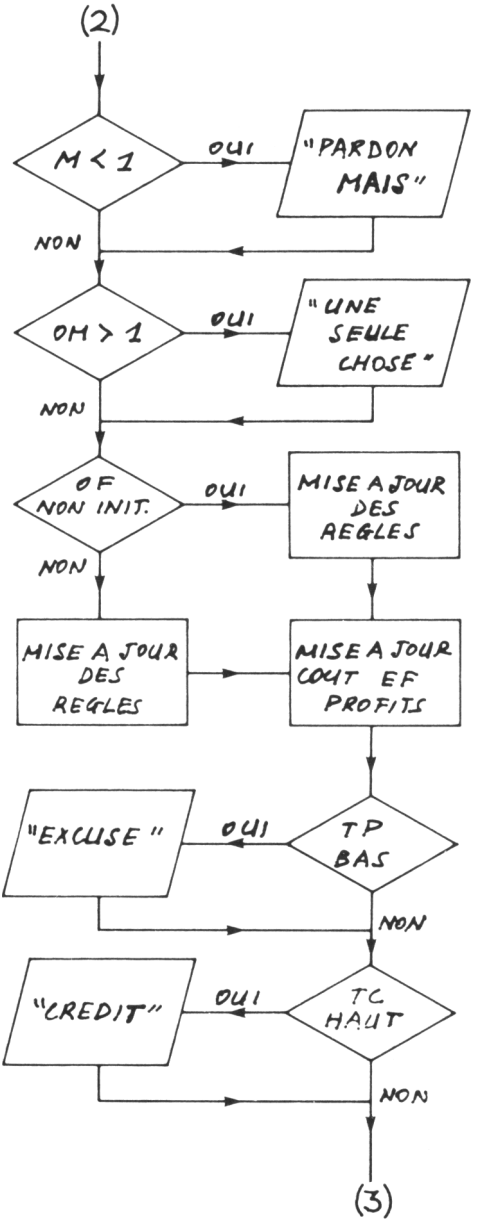
NON, MAIS JE VEUX UN BON SON



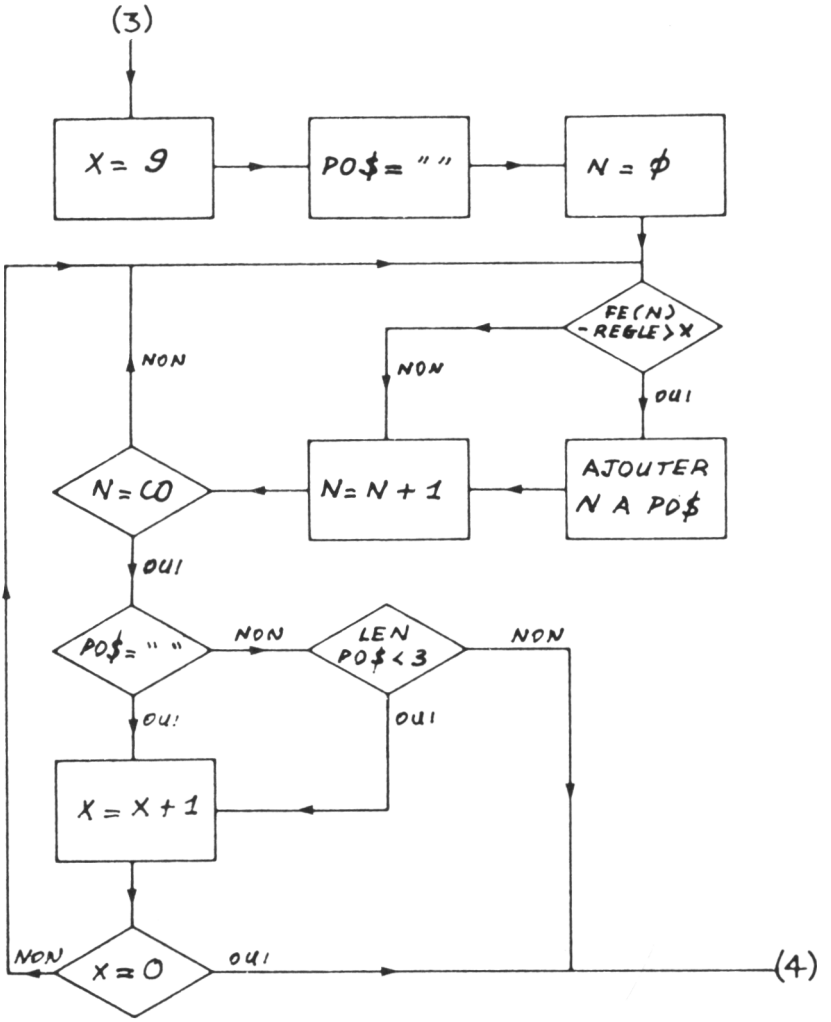
Organigramme 12.1 (a) : un vendeur expert .



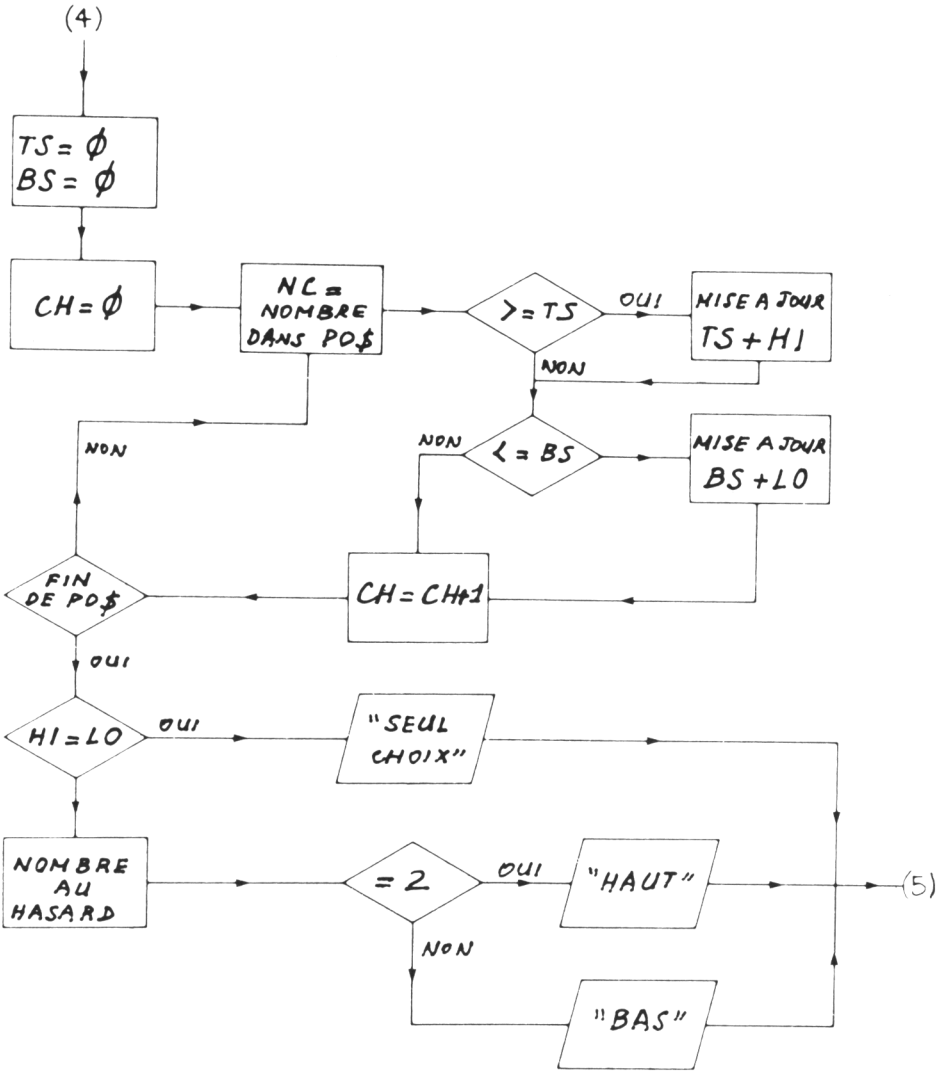
Organigramme 12.1 (b) : un vendeur expert .



Organigramme 12.1 (c) : un vendeur expert .



Organigramme 12.1 (d) : un vendeur expert .



Organigramme 12.1 (e) : un vendeur expert .

Alors le programme retranche 1 pour la partie sur le graphisme et ajoute 2 pour la partie sur le son.

Si le programme ne trouve pas de mot-clé, il redemande une entrée en affichant le message suivant:

PARDON, EXCUSEZ-MOI MAIS

Le programme ne peut traiter qu'une seule chose à la fois, si vous demandez par exemple à la fois le GRAPHISME et le SON, il répondra:

ATTENDEZ, UNE SEULE CHOSE A LA FOIS.

Il est cependant possible de faire des commentaires sur un critère pour lequel on ne vous pose pas de question; dans ce cas, la mise à jour s'effectue quand même correctement (comme avec MAIS).

DECISIONS

Une fois que l'entrée a été décodée, nous évoluons le plus rapidement possible vers la PROCedure de mise à jour. En plus du tableau des désirs, il y en a deux autres qui s'y rattachent. Le premier est celui du COUT, qui donne des indications sur le coût d'un critère particulier, et le second est celui du PROFIT qui indique au vendeur quel appareil il doit particulièrement pousser. Les valeurs pour ces deux tableaux sont obtenues en multipliant le contenu du tableau par les facteurs contenus dans les lignes DATA (voir Tableau 12.2).

Exemple: DE L'EXPANSIBILITE,2,9

indique que le coût de l'EXPANSIBILITE (2) est assez bas, mais qu'il est assez profitable.

NO.	CRITERES	COUT	PROFIT
1	BON BASIC	5	2
2	GRAPHISME	7	2
3	SON	6	2
4	BON CLAVIER	4	2
5	TOUCHES F1,F2	1	5
6	BEAUCOUP DE MEMOIRE	3	6
7	UNE INTERFACE K7	2	2
8	MACRODRIVES	2	4
9	DISQUETTES	5	8
10	BONS LOGICIELS	0	9
11	PORT POUR CARTOUCHES	1	6
12	PORTS JOYSTICKS	1	7
13	ASSEMBLEUR	2	1
14	PORT CENTRONICS	2	5
15	PORT RS-232-C	2	6
16	EXPANSIBILITE	2	9
17	RESEAU LOCAL	3	4
18	CPU 16 BITS	1	7
19	MULTITACHE	5	5
20	BON SERVICE	1	9

Tableau 12.2: Coûts et profits des critères.

Après chaque entrée, le vendeur examine les requêtes que vous avez formulées. En premier lieu il vérifie si vos exigences n'exèdent pas votre crédit, et si c'est le cas, affiche un certain nombre de remarques caustiques à ce propos:

JE NE PENSE PAS QUE VOUS PUISSIEZ VOUS OFFRIR UN TEL LUXE

Il regarde aussi si vos exigences et son profit correspondent, et si ce n'est pas le cas commence à se désintéresser de l'affaire en manifestant de la sorte:

J'AI UN SANGLIER SUR LE FEU

ON FERME A SIX HEURES

Réciproquement,il sera des plus attentionnés en ce qui concerne les ordinateurs répondant à

vos exigences, la PROCedure SAISIS_COMPUTER, fournit une liste des ordinateurs qui correspondent à ce que vous recherchez. La description est la suivante (Tableau 12.3):

NOM	1	2	3	4	5	6	7	8	9	10
IDM PC	7	8	8	9	8	8	8	0	9	9
PAPE TOBISHA	6	7	6	8	8	8	8	0	8	8
SINCLEAR KL	9	9	9	7	7	8	8	9	9	6
ACHRON ILLUSION	8	7	6	6	0	3	7	0	5	5
BANANE IIE	3	5	2	5	0	4	6	0	3	0
SI ELITE	8	8	8	7	7	8	8	0	7	2
COLECTOVISION	5	5	5	5	2	5	5	5	5	1
CANDY COMPUTER	7	6	4	2	0	2	7	0	4	9
COMMANDANT 64	2	8	9	7	7	6	5	0	6	9
ATRIA 600GT	1	8	8	5	0	2	5	0	7	7

NOM	11	12	13	14	15	16	17	18	19	20
IDM PC	7	7	0	7	6	8	8	9	9	9
PAPE TOBISHA	0	0	0	7	6	8	8	9	9	7
SINCLEAR KL	7	7	0	7	6	7	9	9	9	1
ACHRON ILLUSION	0	0	6	0	0	4	1	0	0	2
BANANE IIE	3	5	0	0	6	7	0	0	0	4
SI ELITE	7	4	0	0	6	0	0	0	0	0
COLECTOVISION	7	7	0	0	6	5	0	9	0	0
CANDY COMPUTER	8	7	0	0	6	3	0	0	0	6
COMMANDANT 64	6	7	0	0	2	2	0	0	0	6
ATRIA 600GT	7	7	0	0	6	6	0	0	0	5

(Nom, valeur du critère 1, valeur du critère 2, valeur du critère 3, etc.)

Tableau 12.3: Caractéristiques.

La machine la meilleure sera toujours choisie en premier mais si cela est possible, au moins trois machines (avec de moins bonnes qualités) seront sélectionnées et c'est de ces trois machines que sortira le choix. Le plus cher ou le moins cher seront sélectionnés (au hasard), par exemple:

SI VOUS VOULEZ UNE ROLLS-ROYCE JETEZ UN OEIL AU...

et

SI VOUS VOULEZ DU PREMIER CHOIX PRENEZ-LE...

Si une seule machine correspond, la réponse sera:

UN CHOIX BON MARCHE SERAIT LE...

```
10 ECRAN
20 TITRE
30 INIT
40 REPeat QUESTION
50 SAISIS_QUESTION
60 IF TROUVE_slash THEN
80 TROUV_et
90 END IF
100 IF trouve_ast THEN
110 JOIN_1
120 ELSE
130 JOIN_2
140 END IF
150 PRINT #3,PH$ & "?"
160 INPUT #0, CHAINE$
170 CHAINE$=" " & CHAINE$
180 LD%=1
190 OF%=-1
200 FS%=1
210 NP%=0
220 RU%=0
230 M%=0
240 OM%=0
250 S1%=0
260 S2%=0
270 IF TROUVE_virg THEN
280 ET_ou_MAIS
290 CHAINE$=CHAINE$(CM%+1 TO)
300 END IF
310 OUI_PRESENT
320 NON_PRESENT
330 PAS_PRESENT
340 DOUBLE_NEGATION
350 AIME
360 AIME_PAS
370 OBJETS
380 ADVERBES
390 ADVERBES_NEGATIFS
400 ADJECTIFS
410 ADJECTIFS_NEGATIFS
```

```
420 CHER_BONMARCHE
430 CLS #1
440 IF M%<1 THEN PRINT "PARDON, EXCUSEZ MOI..
. MAIS" : NEXT QUESTION
450 IF OM%>1 THEN PRINT "ATTENDEZ - UNE SEULE
 CHOSE A LA FOIS" : NEXT QUESTION
460 MISE_A_JOUR
470 COUT_PROFIT
480 DEPENSE
490 SAISIS_COMPUTER
500 Q%=Q%+1
510 IF Q%>19 THEN Q%=0
520 END REPeat QUESTION
1000 DEFine PROCedure SAISIS_QUESTION
1010 PH%=RND(0 TO QP%)
1020 PH$=QP$(PH%)
1030 END DEFine SAISIS_QUESTION
2000 DEFine FuNction TROUVE_slash
2010 IS%="/" INSTR PH$
2020 RETurn IS%
2030 END DEFine TROUVE_slash
4000 DEFine PROCedure TROUV_et
4010 IF Q$(Q%,1)="&" THEN
4020 PH$=PH$(1 TO IS%-1) & "EST-CE" & PH$(IS%
 TO)
4030 END IF
4040 END DEFine TROUVE_et
5000 DEFine FuNction trouve_ast
5010 IS%="*" INSTR PH$
5020 RETurn IS%
5030 END DEFine trouve_ast
6000 DEFine PROCedure JOIN_1
6010 PH$=Q$(Q%)(2 TO LEN(Q$(Q%))) & " " & PH$
(1 TO IS%-2) &PH$(IS%+1 TO)
6020 END DEFine JOIN_1
7000 DEFine PROCedure JOIN_2
7010 PH$=PH$ & " " & Q$(Q%)(2 TO LEN(Q$(Q%))
)
7020 END DEFine JOIN_2
8000 DEFine FuNction TROUVE_virg
8010 CM%="," INSTR CHAINE$
8020 RETurn CM%
8030 END DEFine TROUVE_virg
9000 DEFine PROCedure ET_ou_MAIS
9010 IF TROUVE_et + TROUVE_MAIS THEN
9020 IF CHAINE$(1 TO 4)=" NON" THEN
9030 R(Q%)=((R(Q%)*3)-3)/3
```

```
9040 IC(Q%)=((IC(Q%)*3)-(CR(Q%)*3))/3
9050 IP(Q%)=((IP(Q%)*3)-(PR(Q%)*3))/3
9060 ELSE
9070 R(Q%)=R(Q%)+1
9080 IC(Q%)=((IC(Q%)*3)+(CR(Q%)*3))/3
9090 IP(Q%)=((IP(Q%)*3)+(PR(Q%)*3))/3
9100 END IF
9110 END IF
9120 END DEFine ET_ou_MAIS
10000 DEFine FuNction TROUVE_et
10010 S1%="ET" INSTR CHAINE$
10020 RETurn S1%
10030 END DEFine TROUVE_et
11000 DEFine FuNction TROUVE_MAIS
11010 S2%="MAIS" INSTR CHAINE$
11020 RETurn S2%
11030 END DEFine TROUVE_MAIS
12000 DEFine PROCedure OUI_PRESENT
12010 ST%=1
12020 I1$=CHAINE$
12030 REPeat OUI
12040 I1$=I1$(ST% TO)
12050 SP%="OUI" INSTR I1$
12060 IF SP%=0 THEN RETurn
12070 RU%=RU%+1
12080 LD%=1
12090 M%=1
12100 ST%=SP%+1
12110 END REPeat OUI
12120 END DEFine OUI_PRESENT
13000 DEFine PROCedure NON_PRESENT
13010 ST%=1
13020 I1$=CHAINE$
13030 REPeat NON
13040 I1$=I1$(ST% TO )
13050 SP%="NON" INSTR I1$
13060 IF SP%=0 THEN RETurn
13070 LD%=-1
13080 M%=1
13090 ST%=SP%+1
13100 NP%=NP%+1
13110 END REPeat NON
13120 END DEFine NON_PRESENT
14000 DEFine PROCedure PAS_PRESENT
14010 ST%=1
14020 I1$=CHAINE$
14030 REPeat PAS
```

```
14040 I1$=I1$(ST% TO )
14050 SP%="PAS" INSTR I1$
14060 IF SP%=0 THEN RETurn
14070 LD%=-1
14080 M%=1
14090 ST%=SP%+1
14100 NP%=NP%+1
14110 END REPeat PAS
14120 END DEFine PAS_PRESENT
15000 DEFine PROCedure DOUBLE_NEGATION
15010 IF NP%=0 THEN RETurn
15020 IF NP% MOD 2 THEN
15030 RU%=((RU%*3)-3)/3
15040 LD%=-1
15050 ELSE
15060 RU%=RU%+1
15070 LD%=1
15080 END IF
15090 END DEFine DOUBLE_NEGATION
16000 DEFine PROCedure AIME
16010 FOR N=0 TO LI%
16020 SP%=LI$(N) INSTR CHAINE$
16030 IF SP%>0 THEN
16040 IF CHAINE$(SP%-1)=" " THEN LD%=ABS(LD%)
 : M%=1
16050 END IF
16060 END FOR N
16070 END DEFine AIME
17000 DEFine PROCedure AIME_PAS
17010 FOR N=0 TO DL%
17020 SP%=DL$(N) INSTR CHAINE$
17030 IF SP%>0 THEN
17040 IF CHAINE$(SP%-1)=" " THEN LD%=LD%*-1 :
 M%=1
17050 END IF
17060 END FOR N
17070 END DEFine AIME_PAS
18000 DEFine PROCedure OBJETS
18010 FOR N=0 TO OB%
18020 SP%=OB$(N) INSTR CHAINE$
18030 IF SP%>0 THEN
18040 IF CHAINE$(SP%-1)=" " THEN RU%=((RU%*3)
+(LD%*3))/3 : OF%=N : M%=1 : OM%=OM%+1
18050 END IF
18060 END FOR N
18070 END DEFine OBJETS
19000 DEFine PROCedure ADVERBES
```

```
19010 FOR N=0 TO AV%
19020 SP%=AV$(N) INSTR CHAINE$
19030 IF SP%>0  THEN
19040 IF CHAINE$(SP%-1)=" " THEN RU%=((RU%*3)
+(LD%*3))/3 : M%=1
19050 END IF
19060 END FOR N
19070 END DEFine ADVERBES
20000 DEFine PROCedure ADVERBES_NEGATIFS
20010 FOR N=0 TO NV%
20020 SP%=NV$(N) INSTR CHAINE$
20030 IF SP%>0 THEN
20040 IF CHAINE$(SP%-1)=" " THEN LD%=LD%*-1 :
 RU%=((RU%*3)+(LD%*3))/3 : M%=1
20050 END IF
20060 END FOR N
20070 END DEFine ADVERBES_NEGATIFS
21000 DEFine PROCedure ADJECTIFS
21010 FOR N=0 TO AJ%
21020 SP%=AJ$(N) INSTR CHAINE$
21030 IF SP%>0 THEN
21040 IF CHAINE$(SP%-1)=" " THEN RU%=((RU%*3)
+(LD%*3))/3 : M%=1
21050 END IF
21060 END FOR N
21070 END DEFine ADJECTIFS
22000 DEFine PROCedure ADJECTIFS_NEGATIFS
22010 FOR N=0 TO NJ%
22020 SP%=NJ$(N) INSTR CHAINE$
22030 IF SP%>0 THEN
22040 IF CHAINE$(SP%-1)=" " THEN LD%=LD%*-1 :
 RU%=((RU%*3)+(LD%*3))/3 : M%=1
22050 END IF
22060 END FOR N
22070 END DEFine ADJECTIFS_NEGATIFS
23000 DEFine PROCedure CHER_BONMARCHE
23010 FOR N=0 TO HM%
23020 SP%=HM$(N) INSTR CHAINE$
23030 IF SP%>0 THEN
23040 IF CHAINE$(SP%-1)=" " THEN
23050 XX%=N
23060 IF XX%<2 THEN PRINT "BON MARCHE = CAMEL
OTTE":BEEP 1000,10:PAUSE 50
23070 IF XX%>=2 THEN PRINT Q$(Q%)(2 TO)!"C'ES
T PLUTOT CHER":BEEP 1000,10:PAUSE 50
23080 END IF
23090 END IF
```

```
23100 END FOR N
23110 END DEFine CHER_BONMARCHE
24000 DEFine PROCedure MISE_A_JOUR
24010 IF OF%>-1 THEN
24020 R(OF%)=((R(OF%)*3)+(RU%*3))/3
24030 IC(OF%)=((IC(OF%)*3)+((CR(OF%)*RU%)*3))
/3
24040 IP(OF%)=((IP(OF%)*3)+((PR(OF%)*RU%)*3))
/3
24050 ELSE
24060 R(Q%)=((R(Q%)*3)+(RU%*3))/3
24070 IC(Q%)=((IC(Q%)*3)+((CR(Q%)*RU%)*3))/3
24080 IP(Q%)=((IP(Q%)*3)+((PR(Q%)*RU%)*3))/3
24090 END IF
24100 CLS #2
24110 PRINT #2,"DESIRE"
24120 FOR N=0 TO R%
24130 AT #2,0,N*3+15 : PRINT#2, R(N)
24140 END FOR N
24150 PRINT#2,"COUT"
24160 FOR N=0 TO R%
24170 AT #2,1,N*3+15 : PRINT#2, IC(N)
24180 END FOR N
24190 PRINT#2,"PROFIT"
24200 FOR N=0 TO R%
24210 AT #2,2,N*3+15 : PRINT#2, IP(N)
24220 END FOR N
24230 END DEFine MISE_A_JOUR
25000 DEFine PROCedure COUT_PROFIT
25010 FOR N=0 TO OB%
25020 TC%=((TC%*3)+(IC(N)*3))/3
25030 TP%=((TP%*3)+(IP(N)*3))/3
25040 END FOR N
25050 PAPER #2,0 : INK #2,7 : CSIZE #2,1,0
25060 PRINT #2,"    COUT TOTAL ";TC%;"   PROF
IT TOTAL ";TP%;"  SOLDE BANQUE ";BB%;
25070 PAPER #2,4 : INK #2,0 : CSIZE #2,0,0
25080 END DEFine COUT_PROFIT
26000 DEFine PROCedure DEPENSE
26010 IF TP%<Q%*5 THEN TX%=RND(EX%) : INK #1,
4 : PRINT çEX$(TX%) : INK #1,0
26020 IF TC%>BB%/(Q%+1) THEN PT%=RND(CS%) : I
NK #1,2 : PRINT çCS$(PT%) : INK #1,0
26030 TC%=0
26040 TP%=0
26050 END DEFine DEPENSE
27000 DEFine PROCedure SAISIS_COMPUTER
```

```
27010 FOR X=9 TO 0 STEP -1
27020 PO$=""
27030 FOR N=0 TO CO%
27040 IF ((FE(N,Q%)*3)-(R(Q%)*3))>X THEN PO$=
PO$ & N : M%=N
27050 END FOR N
27060 IF LEN(PO$)<3 THEN END FOR X
27070 TS%=0
27080 BS%=10
27090 FOR CH=0 TO LEN(PO$)-1
27100 NC%=PO$(CH+1)
27110 IF C(NC%)>=TS% THEN TS%=C(NC%) : HI%=NC
%
27120 IF C(NC%)<=BS% THEN BS%=C(NC%) : LO%=NC
%
27130 END FOR CH
27140 IF HI%=LO% THEN
27150 PRINT #1,"VOTRE SEULE OPTION EST LE"
27160 CSIZE #1,2,1 : PRINT #1," ***** ";CO$(H
I%);" ***** "; : CSIZE #1,2,0
27170 RETurn
27180 END IF
27190 FI$=CO$(HI%)
27200 LA$=CO$(LO%)
27210 SE%=RND(1 TO 2)
27220 SL%=RND(2)
27230 IF SE%<>2 THEN
27240 PRINT #1,HI$(SL%) : CSIZE #1,2,1 : PRIN
T #1," ***** ";FI$;" *****";: CSIZE #1,2,0
27250 ELSE
27260 PRINT #1,LO$(SL%) : CSIZE #1,2,1 : PRIN
T #1," ***** ";LA$;" *****";: CSIZE #1,2,0
27270 END IF
27280 END DEFine SAISIS_COMPUTER
28000 DEFine PROCedure ECRAN
28010 MODE 4
28020 WINDOW #0,470,40,25,215
28030 BORDER #0,5,4
28040 PAPER #0,0
28050 INK #0,7
28060 CSIZE #0,2,0
28070 CLS #0
28080 WINDOW #1,470,105,25,90
28090 BORDER #1,5,4
28100 PAPER #1,7
28110 INK #1,0
28120 CSIZE #1,2,0
```

```
28130 CLS #1
28140 OPEN #4,SCR_470X25A25X65
28150 BORDER #4,2,2
28160 PAPER #4,2
28170 INK #4,7
28180 CSIZE #4,2,1
28190 CLS #4
28200 WINDOW #2,470,50,25,15
28210 BORDER #2,5,4
28220 PAPER #2,4
28230 INK #2,0
28240 CSIZE #2,0,0
28250 CLS #2
28260 OPEN #3,SCR_470X20A25X195
28270 BORDER #3,5,4
28280 PAPER #3,7
28290 INK #3,0
28300 CSIZE #3,2,0
28310 CLS #3
28320 END DEFine ECRAN
29000 DEFine PROCedure TITRE
29010 PRINT #4," *****    SANA  ELECTRONIQUE
   *****";
29020 END DEFine TITRE
30000 DEFine PROCedure INIT
30010 RESTORE
30020 QP%=5
30030 Q%=19
30040 R%=Q%
30050 OB%=Q%
30060 AJ%=7
30070 AV%=4
30080 LI%=3
30090 DL%=3
30100 NJ%=8
30110 NV%=2
30120 HM%=3
30130 BB%=100
30140 CO%=9
30150 FE%=19
30160 CT%=9
30170 HI%=2
30180 LO%=2
30190 CS%=2
30200 EX%=2
30210 TC%=0
30220 TP%=0
```

```
30230 DIM OB$(OB%,10)
30240 DIM AJ$(AJ%,6)
30250 DIM NJ$(NJ%,7)
30260 DIM AV$(AV%,6)
30270 DIM NV$(NV%,6)
30280 DIM LI$(LI%,7)
30290 DIM DL$(DL%,7)
30300 DIM Q$(Q%,25)
30310 DIM QP$(QP%,16)
30320 DIM HM$(HM%,20)
30330 DIM R(R%)
30340 DIM CR(Q%)
30350 DIM PR(Q%)
30360 DIM IC(Q%)
30370 DIM IP(Q%)
30380 DIM CO$(CO%,30)
30390 DIM FE(CO%,FE%)
30400 DIM DF(CO%,FE%)
30410 DIM C(CT%)
30420 DIM CS$(CS%,100)
30430 DIM EX$(EX%,100)
30440 DIM HI$(HI%,100)
30450 DIM LO$(LO%,100)
30460 DATA "BASIC","GRAPHIQUE","SON","CLAVIER
","FONCTION","MEMOIRE","BANDE","MACRODRIVE","
DISQUE"
30470 DATA "LOGICIEL","CARTRIDGE","JOYSTICK",
"ASSEMBL","CENTRONIC","RS232","EXPAND"
30480 DATA "RESEAU","16-BIT","MULTITACHE","SE
RVICE"
30490 DATA "BON","EXCEL","SUPER","MAGNIF","GE
NIAL","VITE","ESSENT","TOP"
30500 DATA "MAUVAIS","CRAIGN","FAIBL","LENT",
"INEFFIC","PEU","PIRE","MOINDRE","MOINS"
30510 DATA "REAL","TRES","SOUVENT","NECESS","
VRAI"
30520 DATA "JAMAIS","PAS NECES","INUTIL"
30530 DATA "VEUX","AIME","REQUIERRE","EXIGE"
30540 DATA "HAIS","REFUSE","ABBOR","DETEST"
30550 DATA "&UN BON BASIC",5,2,"&DU GRAPHISME
",7,2,"&DU SON",6,2,"&UN BON CLAVIER",4,2
30560 DATA "&DES TOUCHES F1, F2..",1,5,"&BEAU
COUP DE MEMOIRE",3,6,"&UNE INTERFACE K7",2,2
30570 DATA "&DES MACRODRIVES",2,4,"&DES DISQU
ETTES",5,8,"&DES BONS LOGITIELS",0,9
30580 DATA "&PORT POUR CARTOUCHES",1,6,"&DES
PORTS JOYSTICK",1,7,"&UN ASSEMBLEUR",2,1
```

```
30590 DATA "&UN PORT CENTRONICS",2,5,"&UN POR
T RS232",2,6,"&DE L'EXPANDABILITE",2,9,"&UN R
ESEAU LOCAL",3,4
30600 DATA "&UN CPU 16-BIT",1,7,"&DE LA MULTI
TACHE",5,5,"&UN BON SERVICE",1,9
30610 DATA "AIMERIEZ VOUS","POURQUOI PAS","ET
","VOULEZ-VOUS","DESIREZ VOUS","/* IMPORTANT"
30620 DATA "BON MARCHE","PAS CHER"
30630 DATA "CHER","COUTEUX"
30640 DATA "IDM PC",7,8,8,9,8,8,8,0,9,9,7,7,0
,7,6,8,8,9,9,9
30650 DATA "PAPE TOBISHA",6,7,6,8,8,8,8,0,8,8
,0,0,0,7,6,8,8,9,9,7
30660 DATA "SINCLEAR KL",9,9,9,7,7,8,8,9,9,6,
7,7,0,7,6,7,9,9,9,1
30670 DATA "ACHRON ILLUSION",8,7,6,6,0,3,7,0,
5,5,0,0,6,0,0,4,1,0,0,2
30680 DATA "BANANE IIE",3,5,2,5,0,4,6,0,3,0,3
,5,0,0,6,7,0,0,0,4
30690 DATA "SI ELITE",8,8,8,7,7,8,8,0,7,2,7,4
,0,0,6,0,0,0,0,0
30700 DATA "COLECTOVISION",5,5,5,5,2,5,5,5,5,
1,7,7,0,0,6,5,0,9,0,0
30710 DATA "CANDY COLOURED COMPUTER",7,6,4,2,
0,2,7,0,4,9,8,7,0,0,6,3,0,0,0,6
30720 DATA "COMMANDANT 64",2,8,9,7,7,6,5,0,6,
9,6,7,0,0,2,2,0,0,0,6
30730 DATA "ATRIA 600GT",1,8,8,5,0,2,5,0,7,7,
7,7,0,0,6,6,0,0,0,5
30740 DATA 10,9,8,7,6,5,4,3,2,1
30750 DATA "J'AI BIEN PEUR QUE VOUS DEPASSIEZ
    VOTRE BUDGET"
30760 DATA "CES SPECIFICATIONS SEMBLENT
      EXEDER VOTRE BUDGET"
30770 DATA "JE NE PENSE PAS QUE VOUS PUISSIEZ
    VOUS OFFRIR UN TEL LUXE"
30780 DATA "EXCUSEZ MOI, MAIS J'ENTENDS
    LE TELEPHONE SONNER"
30790 DATA "J'AI UN SANGLIER SUR LE FEU"
30800 DATA "ON FERME A SIX HEURES"
30810 DATA "POURQUOI PAS LE ","UN CHOIX BON M
ARCHE SERAIT LE "
30820 DATA "UN BON RAPPORT QUALITE/PRIX POUR
LE"
30830 DATA "SI VOULEZ DU PREMIER CHOIX
    PRENEZ LE"
30840 DATA "DANS L'ETAT ACTUEL DE LA TECHNLOG
```

```
IE, ON NE PEUT BATTRE LE"
30850 DATA "SI VOUS VOULEZ UNE ROLLS ROYCE JE
TEZ UN OEIL AU"
30860 FOR N=0 TO OB%
30870 READ OB$(N)
30880 END FOR N
30890 FOR N=0 TO AJ%
30900 READ AJ$(N)
30910 END FOR N
30920 FOR N=0 TO NJ%
30930 READ NJ$(N)
30940 END FOR N
30950 FOR N=0 TO AV%
30960 READ AV$(N)
30970 END FOR N
30980 FOR N=0 TO NV%
30990 READ NV$(N)
31000 END FOR N
31010 FOR N=0 TO LI%
31020 READ LI$(N)
31030 END FOR N
31040 FOR N=0 TO DL%
31050 READ DL$(N)
31060 END FOR N
31070 FOR N=0 TO Q%
31080 READ Q$(N)
31090 READ CR(N)
31100 READ PR(N)
31110 END FOR N
31120 FOR N=0 TO QP%
31130 READ QP$(N)
31140 END FOR N
31150 FOR N=0 TO HM%
31160 READ HM$(N)
31170 END FOR N
31180 FOR N=0 TO CO%
31190 READ CO$(N)
31200 FOR M=0 TO FE%
31210 READ FE(N,M)
31220 END FOR M
31230 END FOR N
31240 FOR N=0 TO CT%
31250 READ C(N)
31260 END FOR N
31270 FOR N=0 TO CS%
31280 READ CS$(N)
31290 END FOR N
```

```
31300 FOR N=0 TO EX%
31310 READ EX$(N)
31320 END FOR N
31330 FOR N=0 TO LO%
31340 READ LO$(N)
31350 END FOR N
31360 FOR N=0 TO HI%
31370 READ HI$(N)
31380 END FOR N
31390 Q%=0
32000 END DEFine INIT
32010 DEFine PROCedure XX
32020 WINDOW #2,470,240,25,15
32030 PAPER #2,7
32040 INK #2,0
32050 CLS #2
32060 END DEFine XX
```

LA BIBLIOTHÈQUE EDIMICRO

- ***Collection « Ordinateurs professionnels »***

APPLE

de Merly ..... **GUIDE DE L'APPLE**
Tome 1 : l'Apple standard
Tome 2 : les extensions
Tome 3 : les applications

MACINTOSH

Gaucherand ..... **MACINTOSH : outils, progiciels, applications**

SINCLAIR QL

Tenin, Van Thong ..... **Guide pratique du Sinclair QL**
K et S Brain ..... **Intelligence artificielle sur QL**

- ***Collection « Progiciels »***

APPLE

Bonnet, Dinh ..... **Multiplan sur Apple**

IBM PC

Bonnet, Dinh ..... **Lotus 1-2-3 à votre portée**
Bonnet, Dinh ..... **Multiplan sur IBM PC**

MACINTOSH

Bouilloux ..... **Multiplan et Chart sur Macintosh**

GENERAL

Bonnet, Dinh ..... **Mémento Multiplan**

- ***Collection « Langages »***

Gaucherand, Lamoitier ..... **Fichiers en Basic par l'exemple**
Perbost, Berthet ..... **Introduction à MSX**

- ***Collection « Ouvrages de base »***

Lamoitier ..... **Le Traducteur Micro**
Darnis, Van Thong ..... **Graphisme et CAO**
Avon ..... **Courbes de maths en Basic**

- ***Collection « Ordinateurs familiaux »***

VG 5000

Amsler, Bardon ..... **Guide du VG 5000 Philips**
Amsler, Villemaud ..... **Jeux sur VG 5000 Philips**

MO5

Bieber, Perbost, Renucci ..... **Tout sur le MO5**
Perbost, Renucci ..... **Jeux sur MO5**

AMSTRAD

Penfold ..... **L'Amstrad avec plaisir**

ATMOS/ORIC 1

Kosniowsky ..... **Nouveaux Jeux sur ATMOS**
Chane-Hune, Darbois ..... **Guide de l'Oric**
Bayvejiel ..... **Jeux Graphiques sur ATMOS**
Viguier ..... **Premiers Pas en Programmation sur Oric**

SPECTRUM

Bridge, Carnell ..... **Aventures sur Spectrum**
Hurley ..... **Jeux Graphiques sur Spectrum**

COMMODORE 64

Fleurier, Meiller ..... **Jeux sur Commodore 64 : Jeux d'adresse et de hasard**
Fleurier, Meiller ..... **Jeux sur Commodore 64 : Jeux d'action et de réflexion**

ELECTRON/BBC

Bennani, Chaieb ..... **Graphisme et sons sur Electron et BBC**

PHILIPS C7420 VIDEOPAC +

Bardon, de Merly ..... **Jeux sur Philips C7420 Vidéopac +**

TO7

— ouvrages

Bieber, Perbost, Renucci ..... **Guide du TO7**
Perbost, Renucci ..... **Jeux sur TO7**

— logiciels sur cassette

Perbost, Renucci ..... **4 jeux avec manette sur TO7**
Perbost, Renucci ..... **6 jeux : action et réflexion sur TO7**

GENERAL

de Merly ..... **Ordinateur familial : Que choisir ?**
Djama ..... **Je construis mon premier robot**
Ly ..... **Applications familiales en Basic MSX**

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

Achevé d'imprimer en juillet 1985
sur les presses de l'imprimerie Laballery et Cie — 58500 Clamecy
Dépôt légal : juillet 1985 N° d'imprimeur : 506060

EXTRAITS DE PRESSE

GUIDE DE L'APPLE

« Un des meilleurs sur la place. »
Le Figaro.

« Le livre que nous attendions : complet, clair et pratique. »
Apple France.

MULTIPLAN ET CHART SUR MACINTOSH

« Présente l'incomparable avantage de lier Microsoft Chart et Microsoft Multiplan... »
Bernard Vergnes, Directeur de Microsoft pour l'Europe.

MULTIPLAN SUR APPLE

« Nul doute qu'avec un tel outil vous gagnerez en temps et en efficacité. »
Microsoft France.

« Une bonne introduction... Un bon guide... Un bon investissement. »
Science et Vie Micro.

FICHIERS EN BASIC PAR L'EXEMPLE

« Cet ouvrage est recommandé aux personnes amenées à entreprendre la programmation des fichiers du Basic, pour lesquels il constitue une étude pratique, sérieuse et en profondeur. »
Ordi Magazine.

MULTIPLAN SUR IBM PC

« Ce livre rendra de nombreux services. »
OPC.

« Des modèles de qualité professionnelle. »
L'Ordinateur Personnel.

JEUX GRAPHIQUES SUR SPECTRUM

« Ce livre constitue une véritable bibliothèque de jeux variés. »
Plein Jeu.

JEUX SUR COMMODORE 64

« Cet ouvrage permet aux débutants de s'initier au langage Basic en s'amusant... Un livre d'un bon niveau. »
Tilt.

« Les programmes complets et pas trop longs utilisent bien les possibilités du C64. »
L'Ordinateur Individuel.

ORDINATEUR FAMILIAL : QUE CHOISIR ?

« Ce livre a le mérite de simplifier le choix dans la jungle des boutiques informatique. »
Décision Informatique.

PREMIERS PAS EN PROGRAMMATION

« Les exemples sont clairs et bien choisis, la méthode simple et rationnelle. » **Microric**

« Une approche originale dans la littérature consacrée aux micros. »
Décision Informatique.

GUIDE DU TO7

« Ce guide a le privilège d'être présenté de façon claire, dans un style parfaitement accessible à tous. »
Micro 7.

« Un manuel de référence absolue. »
Le Figaro.

JEUX SUR TO7

« De nombreux conseils et "trucs" pour une programmation rapide et efficace. »
Décision Informatique.

« Un excellent livre d'approche, plein de renseignements utiles. »
Micro 7.

GUIDE PRATIQUE DU SINCLAIR QL

« Une somme impressionnante d'informations, des programmes nombreux et intéressants... Un bon outil d'initiation à la pratique de notre Sinclair QL. »
Direco International-Sinclair.

Nombreux autres titres à paraître. Catalogue sur simple demande.

EDIMICRO 121-127, avenue d'Italie, 75013 Paris

INTELLIGENCE ARTIFICIELLE sur SINCLAIR QL

Keith & Steven Brain

ET SI VOTRE QL DEVENAIT INTELLIGENT ?

Il vous ferait la conversation, il vous donnerait des conseils avisés, il serait tour à tour votre élève et votre professeur...
C'est tout à fait possible avec ce livre, qui ne requiert aucune connaissance préalable du sujet. **Tous les aspects de l'Intelligence Artificielle sont couverts, à partir des premiers principes, et sont illustrés par de nombreux programmes.**

Vous apprendrez :

- à bien utiliser le Super-Basic
- à réaliser des routines « intelligentes » pour **ARCHIVE**, le progiciel de bases de données du QL
- à comprendre pourquoi et comment l'Intelligence Artificielle modifiera profondément notre mode de vie dans les années qui viennent.

Sommaire :

L'intelligence artificielle — L'exécution des ordres — La compréhension du langage naturel — La construction des réponses — Les systèmes-experts — Un système-expert qui enrichit les connaissances — Des comparaisons approchées — La reconnaissance des formes — Un professeur intelligent — Des souris et des hommes — Utilisation intelligente du programme ARCHIVE.

ISBN : 2-904457-36-4

170 F.

Edimicro
121-127, Avenue d'Italie 75013 Paris

Edimicro

INTELLIGENCE ARTIFICIELLE SUR SINCLAIR QL

K&S. BRAIN